SHORT TERM LOAD FORECASTING USING COMPUTATIONAL INTELLIGENCE METHODS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

In

Electronics & Communication Engineering

(Specialization in Telematics and Signal Processing)

By

SANJIB MISHRA

Department of Electronics and Communication Engineering

National Institute Of Technology

Rourkela

2008

SHORT TERM LOAD FORECASTING USING COMPUTATIONAL INTELLIGENCE METHODS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

In

Electronics & Communication Engineering

(Specialization in Telematics and Signal Processing)

By

SANJIB MISHRA

Under the Guidance of

Prof. (Dr.) SARAT KUMAR PATRA

Department of Electronics and Communication Engineering

National Institute Of Technology

Rourkela

2008

National Institute Of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled, **"Short Term Load Forecasting Using Computational Intelligence Methods"** submitted by Sri **Sanjib Mishra** in partial fulfillment of the requirements for the award of Master of Technology Degree in **Electronics & Communication Engineering** with specialization in **"Telematics and Signal Processing"** at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Prof. (Dr.) Sarat Kumar Patra
Head of Department
Dept. of Electronics & Communication Engg.
National Institute of Technology
Rourkela-769008

Date:

ACKNOWLEDGEMENTS

This project is by far the most significant accomplishment in my life and it would be impossible without people who supported me and believed in me.

I would like to extend my gratitude and my sincere thanks to my honorable, esteemed supervisor **Prof. (Dr.) Sarat Kumar Patra,** Head, Department of Electronics and Communication Engineering. He is not only a great academician with deep vision but also most importantly a kind person. I sincerely thank for his exemplary guidance and encouragement. His trust and support inspired me in the most important moments of making right decisions and I am glad to have worked with him.

I want to thank all my teachers **Prof. G.S. Rath**, **Prof. K. K. Mahapatra, Prof. S.K. Behera, Prof. S.K. Meher, & Prof. G. Panda** for providing a solid background for my studies and research thereafter. They have been great sources of inspiration to me and I thank them from the bottom of my heart.

I would like to thank all my friends and especially my classmates, for all the thoughtful and mind stimulating discussions we had, which prompted me to think beyond the obvious. I've enjoyed their companionship so much during my stay at NIT, Rourkela.

I would like to thank all those who made my stay in Rourkela an unforgettable and rewarding experience.

I would like to thank Er. Ravi Narayan Tripathy, then Chief General Manager, OPTCL & Er. Jiten Kumar Pattanaik, then General Manager, OPTCL, who had taken great initiative for sanction of my study leave.

Last but not least I would like to thank my parents, who taught me the value of hard work by their own example. They rendered me enormous support during the whole tenure of my stay in NIT Rourkela.

Sanjib Mishra

To my father Laxminarayan Mishra, my mother Late Shantilata Mishra,
and my wife Felicisima D. Mishra

CONTENTS

ABSTRACT

Load forecasting is very essential to the operation of electricity companies. It enhances the energy-efficient and reliable operation of a power system. This dissertation focuses on study of short term load forecasting using different types of computational intelligence methods. It uses evolutionary algorithms (i.e. Genetic Algorithm, Particle Swarm Optimization, Artificial Immune System), neural networks (i.e. MLPNN, RBFNN, FLANN, ADALIN, MFLNN, WNN, Recurrent NN, Wilcoxon NN), and fuzzy systems (i.e. ANFIS). The developed methods give load forecasts of one hour upto 24 hours in advance. The algorithms and networks were have been demonstrated using simulation studies.

The power sector in Orissa has undergone various structural and organizational changes in recent past. The main focus of all the changes initiated is to make the power system more efficient, economically viable and better service oriented. All these can happen if, among other vital factors, there is a good and accurate system in place for forecasting the load that would be in demand by electricity customers. Such forecasts will be highly useful in proper system planning & operations.

The techniques proposed in this thesis have been simulated using data obtained from State Load Dispatch Centre, Orissa for the duration September – 2006 to August – 2007.

LIST OF FIGURES

LIST OF TABLES

ACRONYMS AND ABBREVIATIONS

STLF	Short Term Load Forecasting
MLPNN	Multi Layer Perceptron Neural Network
RNN	Recurrent Neural Network
FLANN	Functional Link Artificial Neural Network
MFLNN	Multi Feedback Layer Neural Network
BP	Back Propagation
GA	Genetic Algorithm
PSO	Particle Swarm Optimization
AIS	Artificial Immune System
ACO	Ant Colony Optimization
BFO	Bacteria Foraging
MAPE	Mean Average Percentage Error

Chapter 1

INTRODUCTION

1. Introduction

1.1 Motivation

Load forecasting is an important component for power system energy management system. Precise load forecasting helps the electric utility to make unit commitment decisions, reduce spinning reserve capacity and schedule device maintenance plan properly. Besides playing a key role in reducing the generation cost, it is also essential to the reliability of power systems. The system operators use the load forecasting result as a basis of off-line network analysis to determine if the system might be vulnerable. If so, corrective actions should be prepared, such as load shedding, power purchases and bringing peaking units on line.

Since in power systems the next days' power generation must be scheduled everyday, day-ahead short-term load forecasting (STLF) is a necessary daily task for power dispatch. Its accuracy affects the economic operation and reliability of the system greatly. Under prediction of STLF leads to insufficient reserve capacity preparation and in turn, increases the operating cost by using expensive peaking units. On the other hand, over prediction of STLF leads to the unnecessarily large reserve capacity, which is also related to high operating cost.

In spite of the numerous literatures on STLF published since 1960s, the research work in this area is still a challenge to the electrical engineering scholars because of its high complexity. How to estimate the future load with the historical data has remained a difficulty up to now, especially for the load forecasting of holidays, days with extreme weather and other anomalous days. With the recent development of new mathematical, data mining and artificial intelligence tools, it is potentially possible to improve the forecasting result.

With the recent trend of deregulation of electricity markets, STLF has gained more importance and greater challenges. In the market environment, precise forecasting is the basis of electrical energy trade and spot price establishment for the system to gain the minimum electricity purchasing cost. In the real-time dispatch operation, forecasting error causes more purchasing electricity cost or breaking-contract penalty cost to keep the electricity supply and consumption balance. There are also some modifications of STLF models due to the implementation of the electricity market. For example, the demand-side management and volatility of spot markets causes the consumer's active response to the electricity price. This should be considered in the forecasting model in the market environment.

1.2 Objectives

Due to some data measurement and transmission problems, in the historical database there might be some erroneous data, which are far away from their real values. The existence of bad data in historical load curve affects the precision of load forecasting results. One of the objectives of this research work is to find a way to detect the erroneous data, eliminate them and evaluate the real data.

Since precise load forecasting remains a great challenge, the objective of this work is to develop some new and practical models with computational intelligence algorithms. As can be seen from the bibliography, many models have been developed for STLF. From the experimental results the conclusion can be drawn that different methods might outperform the others in different situations, i.e. one method might gain the lowest prediction error for one time point, and another might for another time point. How to choose a good method or the combination of different methods for different situations becomes necessary. This research tries to bring forth the advantages of different computational intelligence methods & develop a comprehensive method of selection to fulfill this goal.

1.3 Thesis Organization Outline and Conventions

Following this chapter the remaining chapters of this thesis can be mainly divided into 2 parts: the neural network based models, evolutionary algorithm based models load forecasting with some proposed methods. The thesis is organized as follows.

Chapter 2 gives an overview of the short-term load forecasting problem. The property of the system load, various forecasting methods, and the difficulty in forecasting are introduced. In chapter 3 neural network models for STLF are discussed. This includes multi-layer perceptron neural network (MLP), functional link artificial neural network (FLANN), wavelet based neural network (WNN), & recurrent neural network (RNN) based models. Evolutionary algorithm like genetic algorithm (GA), particle swarm optimization (PSO), & artificial immune system (AIS) based models are applied to short-term load forecasting, which is explained in detail in chapter 4. The final chapter, chapter 5, summarizes the research work and closes the thesis.

In this thesis the following conventions have been employed unless otherwise stated.

- The number of sample load points of per day is 24, i.e. the sampling interval is 1 hour.
- The daily load data is from Orissa Power Transmission Corporation Limited data has been used.
- Mean Absolute Percentage Error (MAPE) has been used as performance criterion for efficiency of models.
- In Table. 3.2. to 3.5, and 4.1. to 4.5, the total absolute percentage error of forecasted period, number of minimum percentage errors achieved by the model with the corresponding parameters, number of times the prediction is less than actual value, number of times the forecasted error is less than 3 percent and MAPE are delineated in the rows from 1 to 5 respectively.
- For simplicity's sake, term "target load", "target day", and "target time point" are used to represent respectively "the load which is to be forecasted", "the day for which the load is to be forecasted", and "the time point at which the load is to be forecasted", and "point *i*" is used to represent the i^{th} point of the daily load curve.

Chapter 2

SHORT-TERM LOAD FORECASTING & OVERVIEW

2. Short-Term Load Forecasting & Overview

2.1 Characteristics of the Power System Load

The system load is the sum of all the consumers' load at the same time. The objective of system STLF is to forecast the future system load. A good understanding of the system characteristics helps to design reasonable forecasting models and select appropriate models operating in different situations. Various factors that influence the system load behavior, can be classified into the following major categories

- Weather
- Time
- Economy
- Random disturbance

The effects of all these factors are introduced in the remaining part of this section to provide a basic understanding of the load characteristics.

Weather

Weather factors include temperature, humidity, precipitation, wind speed, cloud cover, light intensity and etc. The change of the weather causes the change of consumers' comfort feeling and in turn the usage of some appliances such as space heater, water heater and air conditioner. Weather-sensitive load also includes appliance of agricultural irrigation due to the need of the irrigation for cultivated plants. In the areas where summer and winter have great meteorological difference, the load patterns differ greatly.

Normally the intraday temperatures are the most important weather variables in terms of their effects on the load; hence they are often selected as the independent variables in STLF. Temperatures of the previous days also affect the load profile. For example, continuous high temperature days might lead to heat buildup and in turn a new demand peak. Humidity is also an important factor, because it affects the human being's comfort feeling greatly. People feel hotter in the environment of 35 ^{o}C with 70% relative humidity than in the environment of 37

^{o}C with 50% relative humidity. That's why temperature-humidity index (THI) is sometimes employed as an affecting factor of load forecasting. Furthermore, wind chill index (WCI) is another factor that measures the cold feeling. It is a meaningful topic to select the appropriate weather variables as the inputs of STLF.

Time

Time factors influencing the load at time point of the day, holiday, weekday/weekend property and season property. From the observation of the load curves it can be seen that there are certain rules of the load variation with the time point of the day. For example, the typical load curve of the normal winter weekdays (from Monday to Saturday) of the Orissa Grid is shown in Fig. 2.2, with the sample interval of 1 hour, i.e. there are altogether 24 sample points in one day.

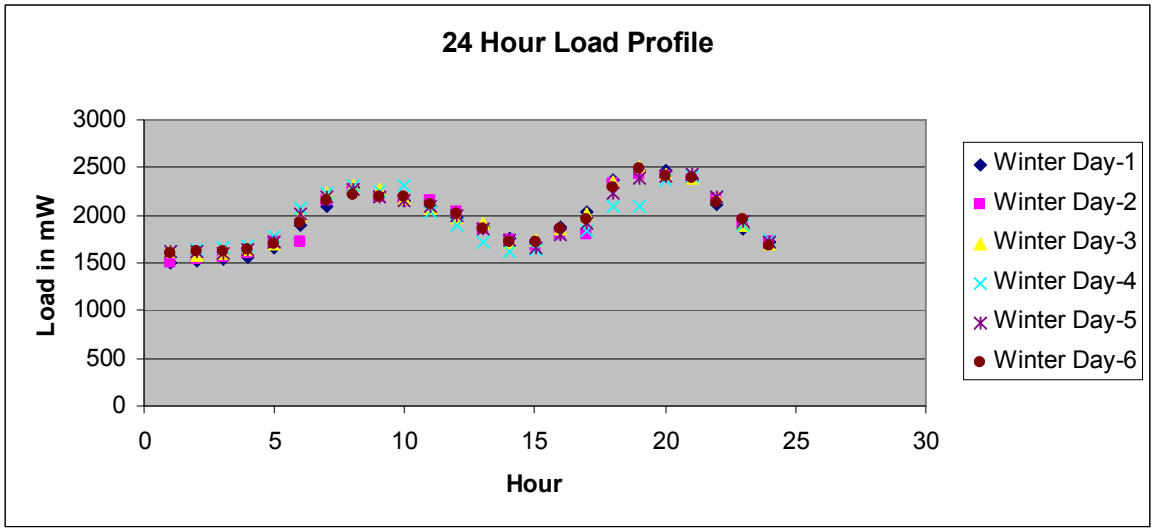


Fig. 2.1. 24 Hour Load Profile of Orissa Grid for a week of December-2006

Typically load is low and stable from 0:00 to 6:00; it rises from around 6:00 to 9:00 and then becomes flat again until around 12:00; then it descends gradually until 17:00; thereafter it rises again until 19:00; it descends again until the end of the day. Actually this load variation with time reflects the people's daily style: working time, leisure time and sleeping time.

There are also some other rules of load variation with time. The weekend or holiday load curve is lower than the weekday curve, due to the decrease of working load. Shifts to and from daylight savings time and start of the school year also contribute to the significant change of the previous load profiles.

Periodicity is another property of the load curve. There is very strong daily, weekly, seasonal and yearly periodicity in the load data. Taking good use of this property can benefit the load forecasting result.

Economy

Electricity is a kind of commodity. The economic situation also influences the utilization of this commodity. Economic factors, such as the degree of industrialization, price of electricity and load management policy have significant impacts on the system load growth/decline trend. With the development of modern electricity markets, the relationship between electricity price and load profile is even stronger. Although time-of-use pricing and demand-side management had arrived before deregulation, the volatility of spot markets and incentives for consumers to adjust loads are potentially of a much greater magnitude. At low prices, elasticity is still negligible, but at times of extreme conditions, price-induced rationing is a much more likely scenario in a deregulated market compared to that under central planning.

Random Disturbance

The modern power system is composed of numerous electricity users. Although it is not possible to predict how each individual user consumes the energy, the amount of the total loads of all the small users shows good statistical rules and in turn, leads to smooth load curves. This is the groundwork of the load forecasting work. But the startup and shutdown of the large loads, such as steel mill, synchrotrons and wind tunnels, always lead to an obvious impulse to the load curve. This is a random disturbance, since for the dispatchers, the startup and shutdown time of these users is quite random, i.e. there is no obvious rule of when and how they get power from the grid. When the data from such a load curve are used in load forecasting training, the impulse component of the load adds to the difficulty of load forecasting. Special events, which are known in advance but whose effect on load is not quite certain, are another source of random disturbance. A typical special event is, for example, a world cup football match, which the dispatchers know for sure will cause increasing usage of television, but cannot best decide the amount of the usage. Other typical events include strikes and the government's compulsory demand-side management due to forecasted electricity shortage.

2.2 Classification of Developed STLF Methods

In terms of lead time, load forecasting is divided into four categories:

- Long-term forecasting with the lead time of more than one year
- Mid-term forecasting with the lead time of one week to one year
- Short-term load forecasting with the lead time of 1 to 168 hours
- Very short-term load forecasting with the lead time shorter than one day

Different categories of forecasting serve for different purposes. In this thesis short-term load forecasting which serves the next day(s) unit commitment and reliability analysis is focused on.

The research approaches of short-term load forecasting can be mainly divided into two categories: statistical methods and artificial intelligence methods [1]. In statistical methods, equations can be obtained showing the relationship between load and its relative factors after training the historical data, while artificial intelligence methods try to imitate human beings' way of thinking and reasoning to get knowledge from the past experience and forecast the future load.

The statistical category includes multiple linear regression [2], stochastic time series [3], general exponential smoothing [4], state space [5], etc. Recently support vector regression (SVR) [6, 7], which is a very promising statistical learning method, has also been applied to short-term load forecasting and has shown good results. Usually statistical methods can predict the load curve of ordinary days very well, but they lack the ability to analyze the load property of holidays and other anomalous days, due to the inflexibility of their structure. Expert system [8], artificial neural network (ANN) [9], fuzzy inference [10], and evolutionary algorithm belong to the computational intelligence category. Expert systems try to get the knowledge of experienced operators and express it in an "if…then" rule, but the difficulty is sometimes the experts' knowledge is intuitive and could not easily be expressed. Artificial neural network doesn't need the expression of the human experience and aims to establish a network between the input data set and the observed outputs. It is good at dealing with the nonlinear relationship between the load and its relative factors, but the shortcoming lies in over fitting and long training time. Fuzzy inference is an extension of expert systems. It constructs an optimal structure of the simplified fuzzy inference that minimizes model

errors and the number of the membership functions to grasp nonlinear behavior of short-term loads, yet it still needs the experts' experience to generate the fuzzy rules. Evolutionary algorithms have been proved to be very useful in multiobjective function optimization, this aspect is used to train the neural network to obtain better results. Generally computational intelligence methods are flexible in finding the relationship between load and its relative factors, especially for the anomalous load forecasting.

Some main STLF methods are introduced as follows.

Regression Methods

Regression is one of most widely used statistical techniques. For load forecasting regression methods are usually employed to model the relationship of load consumption and other factors such as weather, day type and customer class. Engle et al. [11] presented several regression models for the next day load forecasting. Their models incorporate deterministic influences such as holidays, stochastic influences such as average loads, and exogenous influences such as weather. [12 - 15] describe other applications of regression models applied to load forecasting.

Time Series

Time series methods are based on the assumption that the data have an internal structure, such as autocorrelation, trend or seasonal variation. The methods detect and explore such a structure. Time series have been used for decades in such fields as economics, digital signal processing, as well as electric load forecasting. In particular, ARMA (autoregressive moving average), ARIMA (autoregressive integrated moving average) and ARIMAX (autoregressive integrated moving average with exogenous variables) are the most often used classical time series methods. ARMA models are usually used for stationary processes while ARIMA is an extension of ARMA to non stationary processes. ARMA and ARIMA use the time and load as the only input parameters. Since load generally depends on the weather and time of the day, ARIMAX is the most natural tool for load forecasting among the classical time series models.

Fan and McDonald [16] and Cho et al. [17] described implementations of ARIMAX models for load forecasting. Yang et al. [18] used an evolutionary programming (EP) approach to identify the ARMAX model parameters for one day to one week ahead hourly-load-demand-forecasting. The evolutionary programming is a method for simulating evolution and

constitutes a stochastic optimization algorithm. Yang and Huang [19] proposed a fuzzy autoregressive moving average with exogenous input variables (FARMAX) for one day ahead hourly load forecasting.

Neural Networks

The use of artificial neural networks (ANN or simply NN) has been a widely studied load forecasting technique since 1990 [20]. Neural networks are essentially non-linear circuits that have the demonstrated capability to do non-linear curve fitting. The outputs of an artificial neural network are some linear or non-linear mathematical function of its inputs. The inputs may be the outputs of other network elements as well as actual network inputs. In practice network elements are arranged in a relatively small number of connected layers of elements between network inputs and outputs. Feedback paths are sometimes used.

In applying a neural network to load forecasting, one must select one of a number of architectures (e.g. Hopfield, back propagation, Boltzmann machine), the number and connectivity of layers and elements, use of bi-directional or uni-directional links and the number format (e.g. binary or continuous) to be used by inputs and outputs [19].

The most popular artificial neural network architecture for load forecasting is back propagation. This network uses continuously valued functions and supervised learning. That is, under supervised learning, the actual numerical weights assigned to element inputs are determined by matching historical data (such as time and weather) to desired outputs (such as historical loads) in a pre-operational “training session”. Artificial neural networks with unsupervised learning do not require pre-operational training.

Bakirtzis et al. [21] developed an ANN based short-term load forecasting model for the Energy Control Center of the Greek Public Power Corporation. In the development they used a fully connected three-layer feed forward ANN and a back propagation algorithm was used for training. Input variables include historical hourly load data, temperature, and the day of week. The model can forecast load profiles from one to seven days. Also Papalexopoulos et al. [22] developed and implemented a multi-layered feed forward ANN for short-term system load forecasting. In the model three types of variables are used as inputs to the neural networks: seasonal related inputs, weather related inputs, and historical loads. Khotanzad et al [23] described a load forecasting system known as ANNSTLF. It is based on multiple ANN strategy that captures various trends in the data. In the development they used a multilayer perceptron trained with an error back propagation algorithm. ANNSTLF can consider the effect of temperature and relative humidity on the load. It also contains

forecasters that can generate the hourly temperature and relative humidity forecasts needed by the system. An improvement of the above system was described in [24]. In the new generation, ANNSTLF includes two ANN forecasters: one predicts the base load and the other forecasts the change in load. The final forecast is computed by adaptive combination of these forecasts. The effect of humidity and wind speed are considered through a linear transformation of temperature. At the time it was reported in [23], ANNSTLF was being used by 35 utilities across the USA and Canada. Chen et al. [25] also developed a three layer fully connected feed forward neural network and a back propagation algorithm was used as the training method. Their ANN though considers electricity price as one of the main characteristics of the system load. Many published studies use artificial neural networks in conjunction with other forecasting techniques such as time series [26] and fuzzy logic. A recently developed and published recurrent neural network by Savaran [27] is a god bet for the purpose too. The same was applied on STLF and interesting results were found.

Similar Day Approach

This approach [28] is based on searching historical data for days within one, two or three years with similar characteristics to the forecast day. Similar characteristics include weather, day of the week and the date. The load of a similar day is considered as a forecast. Instead of a single similar day load, the forecast can be a linear combination or regression procedure that can include several similar days. The trend coefficients can be used for similar days in the previous years.

Expert Systems

Rule-based forecasting makes use of rules, which are often heuristic in nature, to do accurate forecasting. Expert systems incorporate rules and procedures used by human experts in the field of interest into software that is then able to automatically make forecasts without human assistance.

Ho et al. [29] proposed a knowledge-based expert system for the short-term load forecasting of the Taiwan power system. Operators' knowledge and the hourly observation of system load over the past five years are employed to establish eleven day-types. Weather parameters were also considered. Rahman and Hazim [30] developed a site-independent technique for short-term load forecasting. Knowledge about the load and the factors affecting it is extracted and represented in a parameterized rule base. This rule-based system is complemented by a parameter database that varies from site to site. The technique is tested in different sites in the

United States with low forecasting errors. The load model, the rules and the parameters presented in the paper have been designed using no specific knowledge about any particular site. Results improve if operators at a particular site are consulted.

Fuzzy Logic

Fuzzy logic is a generalization of the usual Boolean logic used for digital circuit design. An input under Boolean logic takes on a value of "True" or "False". Under fuzzy logic an input is associated with certain qualitative ranges. For instance the temperature of a day may be "low", "medium" or "high". Fuzzy logic allows one to logically deduce outputs from fuzzy inputs. In this sense fuzzy logic is one of a number of techniques for mapping inputs to outputs.

Among the advantages of the use of fuzzy logic are the absences of a need for a mathematical model mapping inputs to outputs and the absence of a need for precise inputs. With such generic conditioning rules, properly designed fuzzy logic systems can be very robust when used for forecasting. Of course in many situations an exact output is needed. After the logical processing of fuzzy inputs, a "defuzzification" can be used to produce such precise outputs. [31], [32] and [33] describe applications of fuzzy logic to load forecasting.

Data mining

Data mining is the process that explores information data in a large database to discover rules, knowledge, etc [34, 35]. Hiroyuki Mori et al. proposed a data mining method for discovering STLF rules in [36]. The method is based on a hybrid technique of optimal regression tree and an artificial neural network. It classifies the load range into several classes, and decides which class the forecasted load belongs to according to the classification rules. Then multi layer preceptron (MLP) is used to train the sample in every class. The paper puts an emphasis on clarifying the nonlinear relationship between input and output variables in a prediction model.

Wavelets

A STLF model of wavelet-based networks is proposed [37] to model the highly nonlinear, dynamic behavior of the system loads and to improve the performance of traditional ANNs. The three-layer networks of the wavelet, the weighting, and the summing nodes are built by an evolutionary computing algorithm. Basically, the first layer of wavelet nodes decomposes the input signals into diverse scales of signals, to which different weighting values are given

by the second layer of weighting nodes. Finally the third layer of summing nodes combines the weighted scales of signals into the output. In the evolutionary computing constructive algorithm, the parameters to be tuned in the networks are compiled into a population of vectors. The populations are evolved according to the stochastic procedure of the offspring creation, the competition of the individuals, and the mutation.

To investigate the performance of the proposed evolving wavelet-based networks on load forecasting, the practical load and weather data for the Taiwan power systems were employed. Used as a reference for determining the input variables of the networks, a statistical analysis of correlation functions between the historical load and weather variables was conducted a priori. For comparison, the existing ANNs approach for the STLF, using a back propagation training algorithm, was adopted. The comparison shows wavelet-based ANN forecasting has a more accurate forecasting result and faster speed.

Evolutionary Algorithms

Evolutionary algorithms like genetic algorithm (GA) [38 - 43], particle swarm optimization (PSO) [44 - 46], and artificial immune system (AIS) [47], ant colony optimization (ACO) [48] have been used for training neural networks in short term load forecasting applications. These algorithms are better than back-propagation in convergence and search space capability.

2.3 Requirements of the STLF Process

In nearly all the energy management systems of the modern control centers, there is a short-term load forecasting module. A good STLF system should fulfill the requirement of accuracy, fast speed, automatic bad data detection, friendly interface, automatic data access and automatic forecasting result generation.

Accuracy

The most important requirement of STLF process is its prediction accuracy. As discussed earlier, good accuracy is the basis of economic dispatch, system reliability and electricity markets. The main goal of most STLF literatures and also of this thesis is to make the forecasting result as accurate as possible.

Fast Speed

Employment of the latest historical data and weather forecast data helps to increase the accuracy. When the deadline of the forecasted result is fixed, the longer the runtime of the STLF program is, the earlier historical data and weather forecast data can be employed by the program. Therefore the speed of the forecasting is a basic requirement of the forecasting program. Programs with too long training time should be abandoned and new techniques shortening the training time should be employed. Normally the basic requirement of 24 hour forecasting should be less than 20 minutes.

Automatic Bad Data Detection

In the modern power systems, the measurement devices are located over the system and the measured data are transferred to the control centre by communication lines. Due to the sporadic failure of measurement or communication, sometimes the load data that arrive in the dispatch centre are wrong, but they are still recorded in the historical database. In the early days, the STLF systems relied on the power system operators to identify and get rid of the bad data. The new trend is to let the system itself do this instead of the operators, to decrease their work burden and to increase the detection rate.

Friendly Interface

The interface of the load forecasting should be easy, convenient and practical. The users can easily define what they want to forecast, whether through graphics or tables. The output should also be with the graphical and numerical format, in order that the users can access it easily.

Automatic Data Access

The historical load, weather and other load-relevant data are stored in the database. The STLF system should be able to access it automatically and get the needed data. It should also be able to get the forecasted weather automatically on line, through Internet or through specific communication lines. This helps to decrease the burden of the dispatchers.

Automatic Forecasting Result Generation

To reduce the risk of individual imprecise forecasting, several models are often included in one STLF system. In the past such a system always needs the operators' interference. In other words, the operators have to decide a weight for every model to get the combinative outcome. To be more convenient, the system should generate the final forecasting result according to the forecasting behavior of the historical days.

Portability

Different power systems have different properties of load profiles. Therefore a normal STLF software application is only suitable for the area for which it has been developed. If a general STLF software application, which is portable from one grid to another, can be developed, the effort of developing different software for different areas can be greatly saved. This is a very high-level requirement for the load forecasting, which has not been well realized up until today.

2.4 Difficulties in the STLF

Several difficulties exist in short-term load forecasting. This section introduces them separately.

Precise Hypothesis of the Input-output Relationship

Most of the STLF methods hypothesize a regression function (or a network structure, similar to ANN) to represent the relationship between the input and output variables. How to hypothesize the regression form or the network structure is a major difficulty because it needs detailed a prior knowledge of the problem. If the regression form or the network structure were improperly selected, the prediction result would be unsatisfactory. For example, when a problem itself is a quadratic, the prediction result will be very poor if a linear input-output relationship is supposed. Another similar problem is parameter selection: not only the form of the regression function (or the network structure), but also the parameters of it should be well selected to get a good prediction. Moreover, it is always difficult to select the input variables. Too many or too few input variables would decrease the accuracy of prediction. It should be decided which variables are influential and which are trivial for a certain situation. Trivial ones that do not affect the load behavior should be abandoned.

Because it is hard to represent the input-output relationship in one function, the mode recognition tool, clustering, has been introduced to STLF [49]. It divides the sample data into several clusters. Each cluster has a unique function or network structure to represent the input and output relationship. This method tends to have better forecasting results because it reveals the system property more precisely. But a prior knowledge is still required to do the clustering and determine the regression form (or network structure) for every cluster.

Generalization of Experts' Experience

Many experienced working staff in power grids are good at manual load forecasting. They are even always better than the computer forecasting. So it is very natural to use expert systems and fuzzy inference for load forecasting. But transforming the experts' experience to a rule database is a difficult task, since the experts' forecasting is often intuitive.

The Forecasting of Anomalous Days

Loads of anomalous days are also not easy to be predicted precisely, due to the dissimilar load behavior compared with those of ordinary days during the year, as well as the lack of sufficient samples. These days include public holidays, consecutive holidays, days preceding and following the holidays, days with extreme weather or sudden weather change and special event days. Although the sample number can be greatly enhanced by including the days that are far away from the target day, e.g. the past 5 years historical data can be employed rather than only one or two years, the load growth through the years might lead to dissimilarity of two sample days. From the experimental results it is found that days with sudden weather change are extremely hard to forecast. This sort of day has two kinds of properties: the property of the previous neighboring days and the property of the previous similar days. How to combine these two properties is a challenging task.

Inaccurate or Incomplete Forecasted Weather Data

As weather is a key factor that influences the forecasting result, it is employed in many models. Although the technique of weather forecasting, like the load forecasting, has been improved in the past several decades, sometimes it is still not accurate enough. The inaccurate weather report data employed in the STLF would cause large error.

Another problem is, sometimes the detailed forecasted weather data cannot be provided. The normal one day ahead weather report information includes highest temperature, lowest temperature, average humidity, precipitation probability, maximum wind speed of the day,

weather condition of three period of the day (morning, afternoon and evening). Usually the number of the load forecasting points in a day is 96. If the forecasted weather data of these points can be known in advance, it would greatly increase the precision. However, normal weather reports do not provide such detailed information, especially when the lead time is long. This is a bottleneck of load forecasting.

Less Generalization Ability Caused By Over fitting

Over fitting is a technical problem that needs to be solved for load forecasting. Load forecasting is basically a "training and predicting" problem, which is related to two datasets: training data and testing data. Historical training data are trained in the proposed model and a basic representation can be obtained and in turn used to predict the testing data. For the out coming training module, if the training error for the training data is low but the error for the testing data is high, "over fitting" is said to have occurred. A significant disadvantage of neural networks is over fitting; it shows perfect performance for training data prediction but much poorer performance for the future data prediction. Since the goal of STFL is to predict the future unknown data, technical solutions should be applied to avoid over fitting.

Chapter 3

SHORT TERM LOAD FORECASTING USING ARTIFICIAL NEURAL NETWORK

3. STLF Using Artificial Neural Network

3.1. Multi Layer Perceptron Neural Network

Multi Layer Perceptron is one of the most widely used neural network model today due to its ability to construct a model for complex non-linear relationships between inputs and outputs. The network consists of number of layers each containing number of neuron units and weights associated with the connection between them where the information is passed as feed-forward manner. In our model there are 1 input layer, 1 hidden layer and 1 output layer involved, as this type of MLP is proven to address almost any kind of non-linear relationship. Figure 3.1 shows the architecture of the network used in our model.

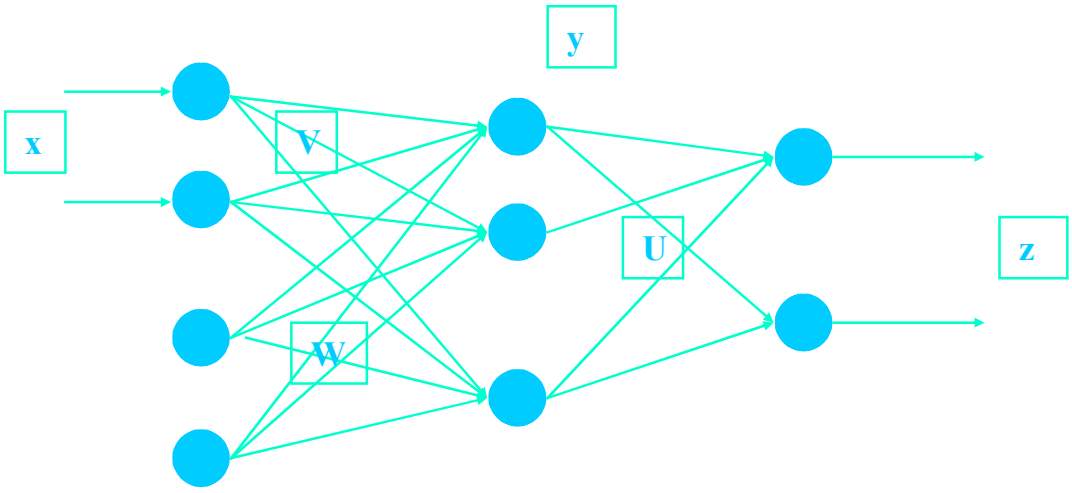


Fig. 3.1. Multi Layer Perceptron Neural Network (MLPNN)

'x' denotes the input vector, 'W' denotes the weight vector between input and hidden layer, 'V' denotes the output of input layer, 'U' denotes the weight vector between hidden and output layer, 'y' denotes the output of hidden layer and 'z denotes the final output. Number of hidden layer neurons and activation function are the parameters with which the network can be fine tuned to get minimum error.

3.1.1 Introduction

A Multi Layer Perceptron Neural Network (MLPNN) can be used for Short Term Load Forecasting (STLF) purpose. The MLPNN is trained to approximate the nonlinear function *F (.)* between the target load (load @ one hour ahead) and the input variables. The MLPNN comprises a layer of input, one or more hidden layer(s) and a layer of output. The MLPNN can be trained by several algorithms like back propagation (BP), genetic algorithm (GA), particle swarm optimization (PSO), artificial immune system (AIS), ant colony optimization (ACO) and bacteria foraging (BFO).

3.1.2 MLPNN Model for Hour Ahead Load Forecasting

The parameters of three layer perceptron artificial neural network for hour ahead load forecasting are as given below:

- No. of layers: 3 (Input layer, Hidden layer, Output layer)
- No of neurons in hidden layer: 15 to 19
- No of neurons in output layer: 1
- Activation function of hidden layer: logsig
- Activation function of output layer: Linear
- Training algorithm: Back-Propagation
- Learning rate (α): 0.1
- Momentum factor (γ): 0.98
- No. of input variables: 9 (same day: hr-1, hr-2, hr-3; day-1: hr, hr-1, hr-2; day-7: hr, hr-1, hr-7)
- No of output variable: 1 (same day: hr)
- No of data sets in each epoch: 63
- No. of epochs for training: 100

The number of neurons in the hidden layer taken was 15, 17 and 19 for testing purpose with "logsig" activation function. The MAPE obtained by using 15, 17 and 19 hidden neurons were found to be 3.217%, 3.1582% and 3.238% respectively. The corresponding results are shown in Fig. 3.1. The forecasted load pattern is shown in dashed line and actual load pattern in solid line. Other activation functions and increment in number of neurons did not

significantly impact the load forecast accuracy. After many experimentation the learning rate (α) and momentum factor (γ) was taken to be 0.1 and 0.98 respectively.

The minimum Mean Average Percentage Error (MAPE) was found to be 3.1582 % by a network with 17 hidden neurons and Logsig activation function (this is shown in Fig. 3.2.(b)).

3.1.3 Simulation Results

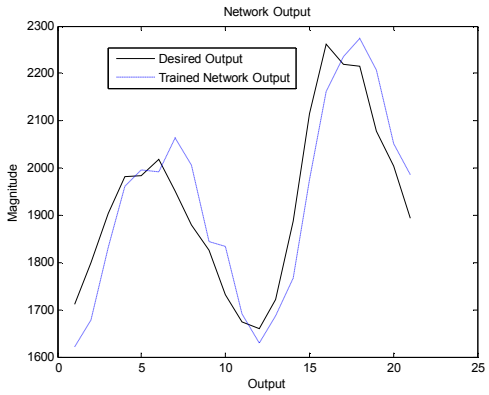


Fig. 3.2. (a) 15 Nos. Hidden Neurons

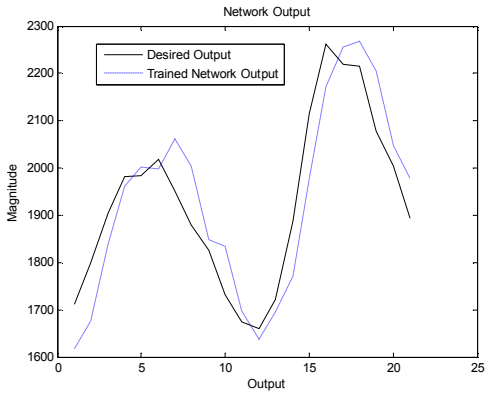


Fig. 3.2. (b) 17 Nos. Hidden Neurons

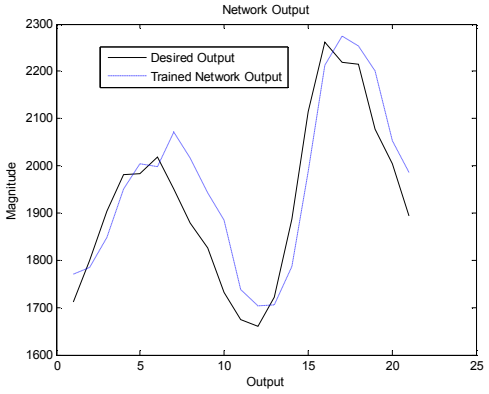


Fig. 3.2. (b) 19 Nos. Hidden Neurons

Fig. 3.2. Hour Ahead Prediction Plot of MLP-BP with Logsig activation function

3.1.4 MLPNN Model for Day Ahead Load Forecasting

The parameters of three layer perceptron based artificial neural network for day ahead load forecasting are as given below:

- No. of layers: 3 (Input layer, Hidden layer, Output layer)
- No of neurons in hidden layer: 4 to 9
- No of neurons in output layer: 24
- Activation function of hidden layer: logsig
- Activation function of output layer: Linear
- Training algorithm: Back-Propagation
- Learning rate (α): 0.1
- Momentum factor (γ): 0.98
- No. of input variables: 24 (same day: 24 hours)
- No of output variable: 24 (next day: 24 hours)
- No of data sets in each Epoch: 20
- No. of Epochs for training: 50

The number of neurons in the hidden layer taken was 4, 7, and 9 for testing purpose with "logsig" activation function. The MAPE obtained by using 4, 7 and 9 hidden neurons were found to be 1.2711%, 1.5412% and 1.4951% respectively. The corresponding results are shown in Fig. 3.2. The forecasted load pattern is shown in dashed line and actual load pattern in solid line. Other activation functions and number of neurons did not significantly impact the load forecast accuracy.

The minimum Mean Average Percentage Error (MAPE) was found to be 1.2711 % by a network with 4 hidden neurons and Logsig activation function (this is shown in Fig. 3.3.(b)).

3.1.5 Simulation Results

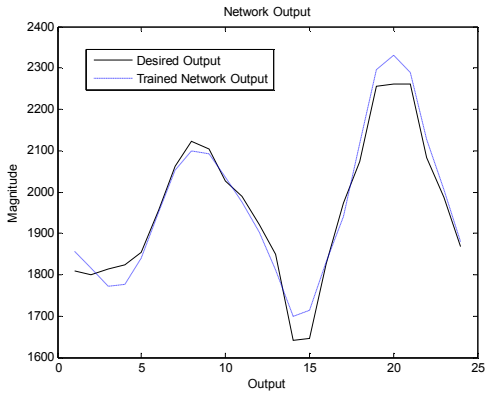


Fig. 3.3.(a) 7 Nos. Hidden Neurons

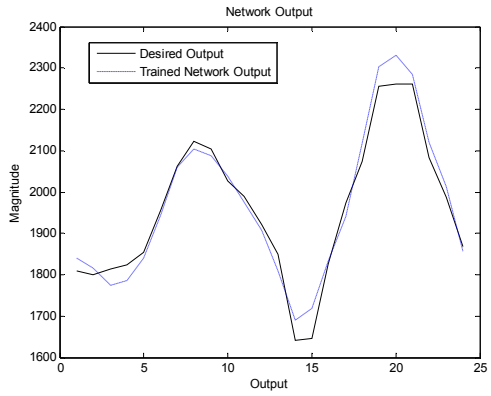


Fig. 3.3.(b) 9 Nos. Hidden Neurons

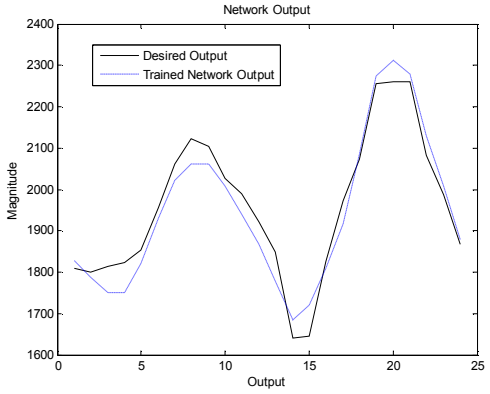


Fig. 3.3.(c) 11 Nos. Hidden Neurons

Fig. 3.3. Day Ahead Prediction Plot of MLP-BP with Logsig activation function

3.2. Recurrent Neural Network

3.2.1 Introduction

Time series nonlinear predictors can be formed by placing zero-memory nonlinearity within the output stage of classical linear predictor. The nonlinearity is restricted to the output stage, as in a single layer neural network realization. On the other hand, if the nonlinearity is distributed through many layers of weighted interconnections, the concept of neural networks

is fully exploited and more powerful nonlinear predictors may ensue. For the purpose of prediction, memory stages may be introduced at the input or within the network. In the prediction of hourly load, the network will have only one output neuron with a predicted value. For a dynamic system, such as a recurrent neural network for prediction, the state represents a set of quantities that summarizes all the information about the past behavior of the system that is needed to uniquely describe its future behavior.

The provision of feedback with delay introduces memory to the network and so is appropriate for prediction in case of recurrent neural networks. The feedback within it can be achieved either a local or global manner. The local feedback is achieved by the introduction of feedback within the hidden layer, whereas the global feedback is produced by the connection of the network output to the network input as shown in figure 1. Inter neuron connections are also possible. The use of the large number of tapped delay feedback input increases the input dimension, resulting in increased dimensionality problem.

Furthermore, the recurrent systems can inherently produce multistep ahead predictions; so, the multistep ahead prediction models, which are required in some process control applications, such as predictive control, can efficiently be built by RNNs [50]. Thus, the RNNs have attracted great interest. The Hopfield [51], the Elman [52], the Jordan [53], the fully recurrent [54], the locally-recurrent [55], the recurrent radial basis function [56], and the block-structured recurrent [57] networks are some of the examples of RNNs. In these structures, the feedback weights, assumed to be unity, are not trainable. The fully recurrent neural network allows any neuron to be connected to any other neuron in the network. While being more general, it lacks stability.

Here the architecture and training procedure of a new RNN useful for short term load prediction / forecasting is presented. The structure of the proposed RNN differs from the other RNNs in the literature. The main difference of the proposed network compared to the available RNNs is that the temporal relations are provided by means of neurons arranged in three feedback layers, not by simple feedback elements, in order to enrich the representation capabilities of the recurrent networks. The feedback signals are processed in three feedback layers which contain neurons as in feedforward layers. In these feedback layers, the weighted sums of the delayed outputs of the hidden and output layers are passed through activation functions and applied to the feedforward neurons via some adjustable weights.

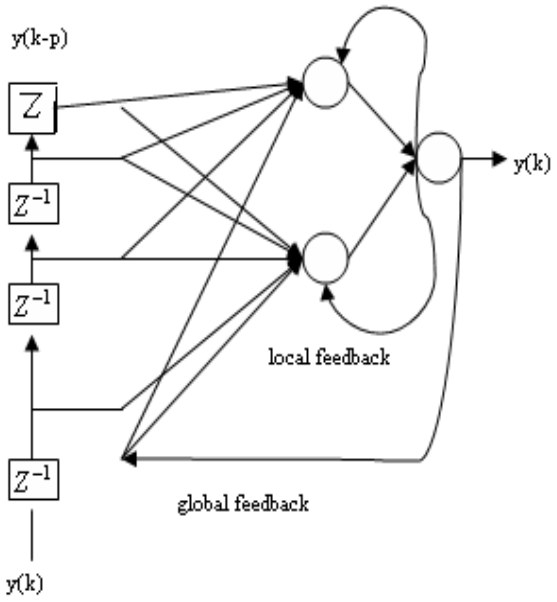


Fig. 3.4. Structure of a recurrent neural network with local and global feedback

3.2.2 Proposed Recurrent Neural Network

The RNN architecture used here is presented in figure 2, where $Input(k) \& y(k)$ represents the input and output of the RNN, respectively, and k is the time index. The RNN has three feed forward and feedback layers. In the feed forward layers, $W_1 \& W_2$, represent the weights between the input and hidden layers, and the hidden and output layers, respectively. In addition to the feed forward layers, the RNN has two local and one global feedback layers. In these feedback layers, the weighted sums of the delayed outputs of the hidden and output layers are applied to certain activation functions as in the feed forward layer neurons. $W_{b1}, W_{b2} \& W_{b3}$ represent the weights connected to the inputs of the feedback layer neurons and z^{-1} represents the time delay operators. The outputs of the feedback layers neurons $h_c(k), y_c(k) \& z_c(k)$ are applied to the hidden and output layers neurons via the adjustable weights $W_{c1}, W_{c2} \& W_{c3}$.

The number of hidden neurons in this case is taken as two but should be tuned as per requirement of individual problem requirements. The number of neurons in the feedback layer from the hidden-to-hidden layer is set equal to the number of the hidden layer neurons i.e. two. The number of neurons in the feedback layer from the output-to-hidden layer is set

equal to the number of the output layer neurons i.e. one. The number of neurons in the feedback layer from the output-to-output layer is set equal to the number of the output layer neurons i.e. one. However, their numbers can be adjusted to improve the accuracy on case to case basis. The numbers were finalized on trial and error.

Since the weights are updated by the back propagation method, the calculation of the Jacobian matrix is required. The backward phase computations from $k = T$ to $k = 1$ are performed by means of the back propagated path values of the MFLNN. When the forward and backward phases of the computations are completed, the sensitivities for each weight, which form the Jacobian matrix, are obtained as in the back propagation algorithm.

As it was expressed previously, the elements of the Jacobian matrix are computed in two stages which are referred to as the forward and backward phases. In the forward phase, the RNN actions are computed and stored from $k = 1$ to $k = T$ through the trajectory. The errors at every k are determined as the differences between the desired outputs and the RNN outputs. The initial values for the output of the hidden layer (h) and of the output layer (y) are set to 0. The net quantities produced at the input of the activation functions of the feedback neurons are

$$h(0) = 0, y(0) = 0$$
$$forout_{ch}(k) = \left[W_{b1} * h(k-1)\right] + B_{b1}$$
$$forout_{cy}(k) = \left[W_{b2} * y(k-1)\right] + B_{b2}$$
$$forout_{cz}(k) = \left[W_{b3} * y(k-1)\right] + B_{b3}$$

Where W_{b1}, W_{b2} & W_{b3} the input weights of the feedback layers and B_{b1}, B_{b2} & B_{b3} are the biases of the feedback layer neurons. The outputs of the feedback layer neurons h_c, y_c & z_c are computed by

$$h_c = \tanh\left(forout_{ch}(k)\right)$$
$$y_c = \tanh\left(forout_{cy}(k)\right)$$
$$z_c = \tanh\left(forout_{cz}(k)\right)$$

Where tanh represent the activation functions of the feedback layer neurons. The net quantities $forout$ of the hidden layer neurons and their outputs (h) are computed by

$$forout_h(k) = \left[W_1 * Input(k)\right]^T + \begin{bmatrix} \left[W_{c1} * h_c(k)\right] + \\ \left[W_{c2} * y_c(k)\right]^T + B_1 \end{bmatrix}$$
$$h(k) = \tanh\left(forout_h(k)\right)$$

Where W_1 represents the weights between the input and hidden layers, and B_1 the biases applied to the hidden layer neurons. W_{c1} & W_{c2} are the weights of the feedback layers. tanh represents the hidden layer activation functions. Similarly, the net quantities $forout_y$ of the output layer neurons and their outputs (y) are computed by

$$forouty(k) = \left[W2^T * h(k)\right] + \left[Wc3^T * zc(k)\right] + B2$$
$$y(k) = purelin\left(forouty(k)\right)$$

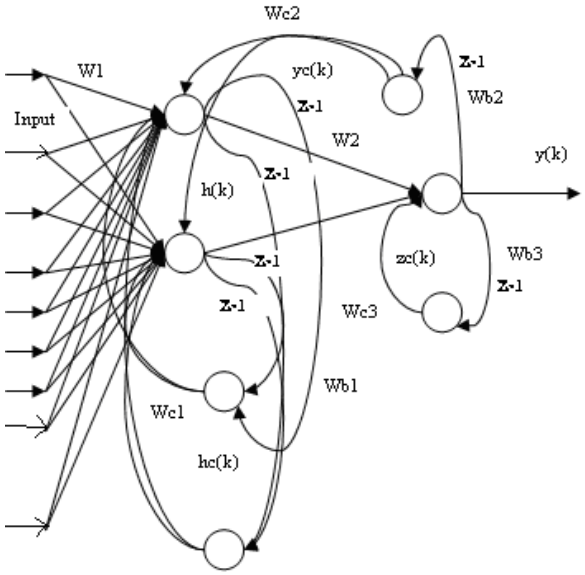


Fig. 3.5. Structure of the proposed RNN

Where W_2, B_2 & tanh represent the weights between the hidden and output layers, the biases applied to the output layer neurons, and the output layer activation functions, respectively. W_{c3} represents the output weights of the feedback layer. The error signal (e) is defined as the difference between the RNN output (y) and the desired output $(Output)$.

$$e(k) = y(k) - Output(k)$$

The weights are adjusted to minimize the error (e), so the sensitivities with respect to each weight have to be computed. At every (k), the sensitivity for each weight is computed by multiplying the input of this weight in the RNN and the back propagated path, so the inputs of the weights in the back propagated path have to be computed. Therefore, after completing the forward phase computations, the backward phase computation is carried out through the

back propagated path of RNN from $k = T$ to $k = 1$. The local sensitivities at $k = T + 1$ are set to 0.

$$\delta_{c3}(T+1) = 0$$
$$\delta_{c2}(T+1) = 0$$
$$\delta_{c1}(T+1) = 0$$

The local sensitivities are obtained as

$$\delta_2(k) = \sec h^2\left(forout_y(k)\right) * \begin{bmatrix} 1 + \left(W_{b2} * \delta_{c2}(k+1)\right) + \\ \left(W_{b3} * \delta_{c3}(k+1)\right) \end{bmatrix}$$
$$\delta_1(k) = \sec h^2\left(forout_h(k)\right) * \begin{bmatrix} \left(W_{b1} * \delta_{c1}(k+1)\right) + \\ \left(W_2 * \delta_2(k)\right) \end{bmatrix}$$
$$\delta_{c3}(k) = \sec h^2\left(forout_{cz}(k)\right) * \left[W_{c3} * \delta_2(k)\right]$$
$$\delta_{c2}(k) = \sec h^2\left(forout_{cy}(k)\right) * \left[W_{c2} * \delta_1(k)\right]$$
$$\delta_{c1}(k) = \sec h^2\left(forout_{ch}(k)\right) * \left[W_{c1} * \delta_1(k)\right]$$

Then, the sensitivity for each weight is computed by multiplying the values scaled by this weight in the RNN and the back propagated path as follows:

$$\frac{\partial e(k)}{\partial W_2} = \delta_2(k) * h \quad (k)$$
$$\frac{\partial e(k)}{\partial B_2} = \delta_2(k)$$
$$\frac{\partial e(k)}{\partial W_1} = \delta_1(k) * Input \quad (k)$$
$$\frac{\partial e(k)}{\partial B_1} = \delta_1(k)$$
$$\frac{\partial e(k)}{\partial W_{c3}} = \delta_2(k) * z_c{}^T(k)$$
$$\frac{\partial e(k)}{\partial W_{b3}} = \delta_{c3}(k) * y^T(k-1)$$
$$\frac{\partial e(k)}{\partial W_{c2}} = \delta_1(k) * y_c{}^T(k)$$
$$\frac{\partial e(k)}{\partial W_{b2}} = \delta_{c2}(k) * y^T(k-1)$$
$$\frac{\partial e(k)}{\partial W_{c1}} = \delta_1(k) * h_c{}^T(k)$$
$$\frac{\partial e(k)}{\partial W_{b1}} = \delta_{c1}(k) * h^T(k-1)$$
$$\frac{\partial e(k)}{\partial B_{b3}} = \delta_{c3}(k)$$
$$\frac{\partial e(k)}{\partial B_{b2}} = \delta_{c2}(k)$$
$$\frac{\partial e(k)}{\partial B_{b1}} = \delta_{c1}(k)$$

Then network weights and biases can be calculated as follows:

$$W_1 = W_1 - \left[\mu * e(c) * \frac{\partial e(k)}{\partial W_1} \right]$$
$$B_1 = B_1 - \left[\mu * e(c) * \frac{\partial e(k)}{\partial B_1} \right]$$
$$W_2 = W_2 - \left[\mu * e(c) * \frac{\partial e(k)}{\partial W_2} \right]$$
$$B_2 = B_2 - \left[\mu * e(c) * \frac{\partial e(k)}{\partial B_2} \right]$$

$$W_{b1} = W_{b1} - \left[\mu * e(c) * \frac{\partial e(k)}{\partial W_{b1}} \right]$$
$$B_{b1} = B_{b1} - \left[\mu * e(c) * \frac{\partial e(k)}{\partial B_{b1}} \right]$$
$$W_{c1} = W_{c1} - \left[\mu * e(c) * \frac{\partial e(k)}{\partial W_{c1}} \right]$$
$$W_{b2} = W_{b2} - \left[\mu * e(c) * \frac{\partial e(k)}{\partial W_{b2}} \right]$$
$$B_{b2} = B_{b2} - \left[\mu * e(c) * \frac{\partial e(k)}{\partial B_{b2}} \right]$$
$$W_{c2} = W_{c2} - \left[\mu * e(c) * \frac{\partial e(k)}{\partial W_{c2}} \right]$$
$$W_{b3} = W_{b3} - \left[\mu * e(c) * \frac{\partial e(k)}{\partial W_{b3}} \right]$$
$$B_{b3} = B_{b3} - \left[\mu * e(c) * \frac{\partial e(k)}{\partial B_{b3}} \right]$$
$$W_{c3} = W_{c3} - \left[\mu * e(c) * \frac{\partial e(k)}{\partial W_{c3}} \right]$$

(μ) is the learning rate of the RNN. "Purelin" and "logsig" activation functions are used in this simulation.

3.2.3 RNN Model for Hour Ahead Load Forecasting

The parameters of the system for next hour load forecasting are as given below:

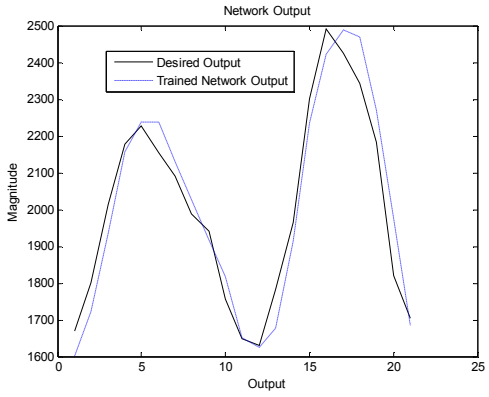
- No of neurons in hidden layer: 2 tor 3
- No of neurons in output layer: 1
- Activation function of hidden layer: tanh
- Activation function of output layer: Linear

- Training algorithm: as proposed by Savaran with modification
- Learning rate (α): 0.18
- No. of input variables: 9 (same day: hr-1, hr-2, hr-3; day-1: hr, hr-1, hr-2; day-7: hr, hr-1, hr-7)
- No of output variable: 1 (same day: hr)
- No of data sets: 63
- Initialization: A.*Rand-(A/2) or 2.*Rand-1 or Rands or Randnr or Randnc or Rand or Nguyen-Widrow
- Cost function: error, instead of half of error square as proposed by Savaran

Tanh activation function was taken for testing purpose. Other activation functions did not significantly impact the load forecast accuracy. Much experimentation were performed with different combination of initialization methods and no. of hidden neurons. The MAPE obtained for different combinations of number of hidden neurons and initialization methods are shown in Table 3.1. and Table 3.2. In Fig. 3.5.(a), 3.5.(b), 3.5.(c), 3.5.(d), 3.5.(e), 3.5.(f), 3.5.(g), 3.5.(h), 3.5.(i), 3.5.(j), 3.5.(k), 3.5.(l), 3.5.(m) and 3.5.(n) the dashed lines are trained network forecasted load pattern and solid lines are actual load pattern.

The minimum Mean Average Percentage Error (MAPE) was found to be 2.9699 % by a network with 2 hidden neurons and A.*Rand-(A/2) initialization method (this is shown in Fig. 3.6.(a)).

3.2.4 Simulation Results



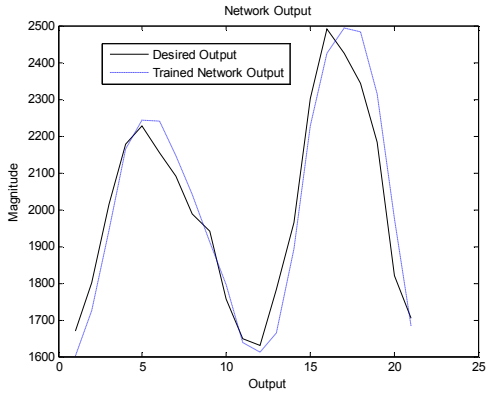


Fig. 3.6.(a) A.*Rand-(A/2) Initialization, 2 Nos. Hidden Neurons **Fig. 3.6.(b) A.*Rand-(A/2) Initialization, 3 Nos. Hidden Neurons**

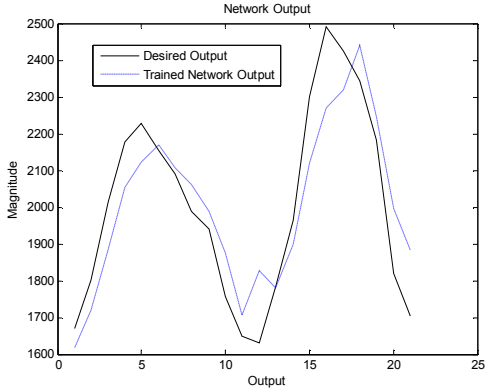


Fig. 3.6.(c) 2.*Rand-1 Initialization, 2 Nos. Hidden Neurons

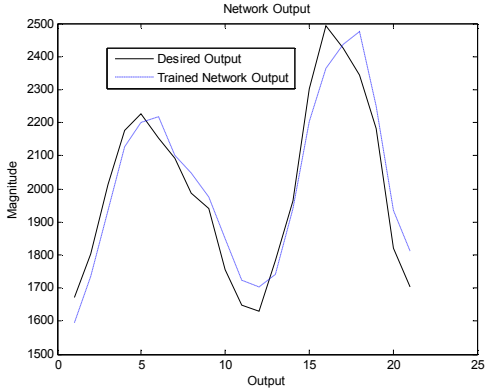


Fig. 3.6.(d) 2.*Rand-1 Initialization, 3 Nos. Hidden Neurons

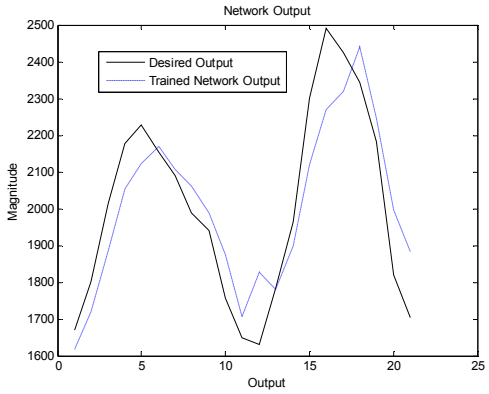


Fig. 3.6.(e) Rands Initialization, 2 Nos. Hidden Neurons

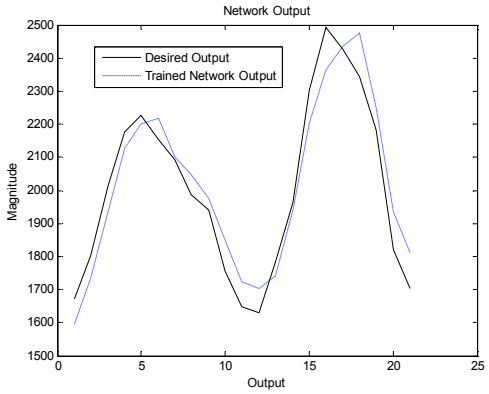


Fig. 3.6.(f) Rands Initialization, 3 Nos. Hidden Neurons

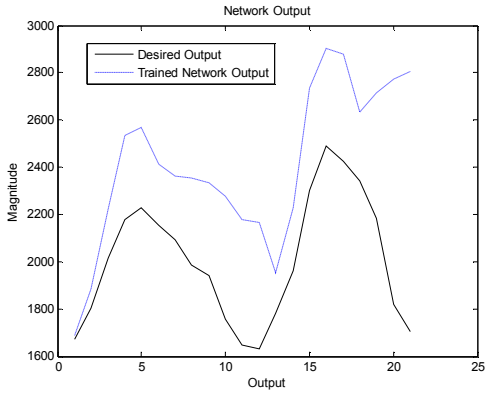


Fig. 3.6.(g) Randnr Initialization, 2 Nos. Hidden Neurons

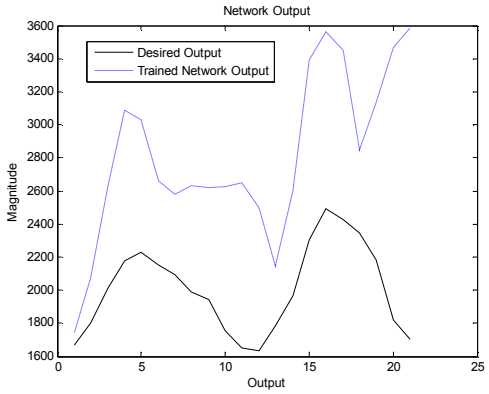


Fig. 3.6.(h) Randnr Initialization, 3 Nos. Hidden Neurons

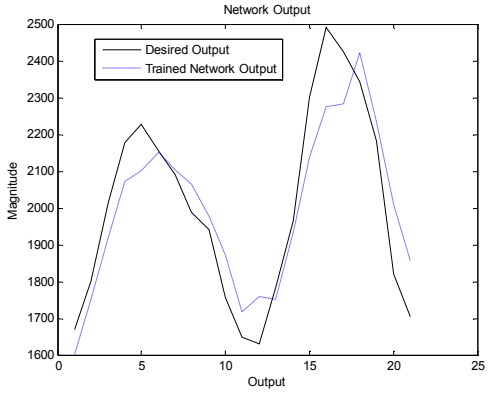


Fig. 3.6.(i) Randnc Initialization, 2 Nos. Hidden Neurons

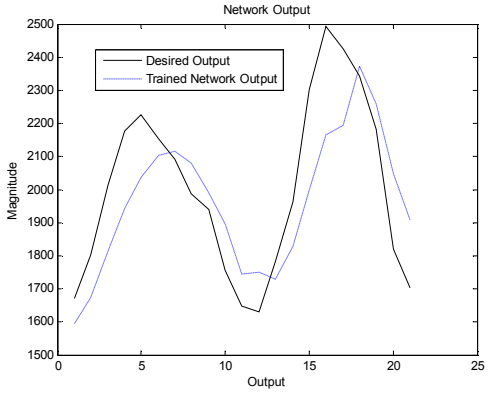


Fig. 3.6.(j) Randnc Initialization, 3 Nos. Hidden Neurons

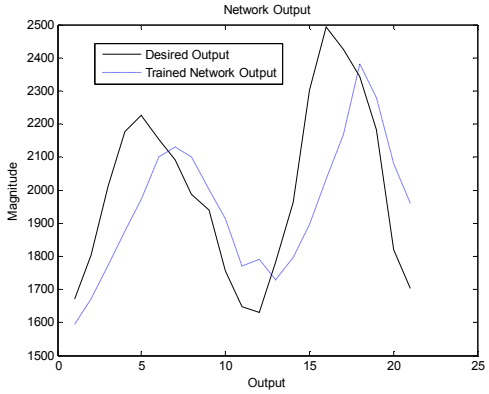


Fig. 3.6.(k) Rand Initialization, 2 Nos. Hidden Neurons

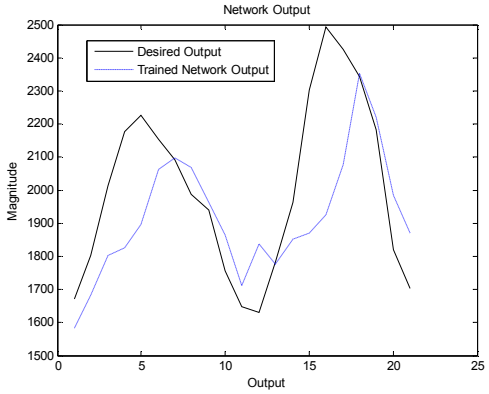


Fig. 3.6.(l) Rand Initialization, 3 Nos. Hidden Neurons

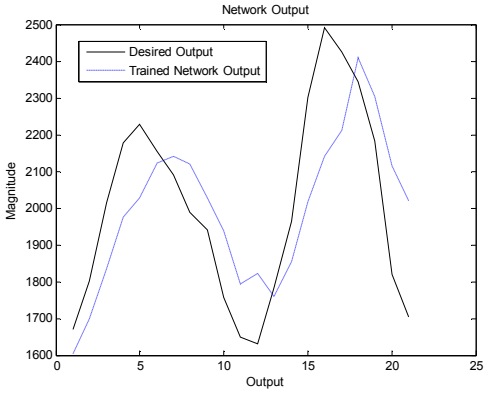


Fig. 3.6.(n) Nguyen-Widrow, 2 Nos. Hidden Neurons

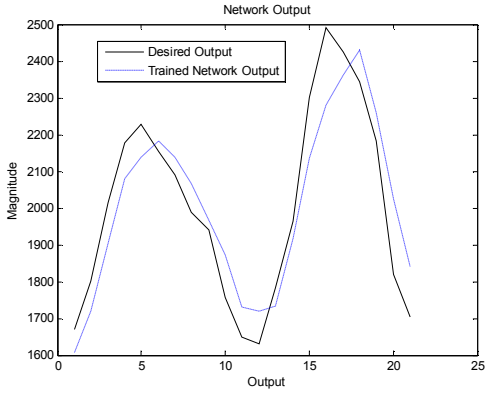


Fig. 3.6.(m) Nguyen-Widrow, 3 Nos. Hidden Neurons

Fig. 3.6. Hour Ahead Prediction by RNN with different Initialization method and hidden neurons

A.*Rand-(A/2) 9-2-1	A.*Rand-(A/2) 9-3-1	2.*Rand-1 9-2-1	2.*Rand-1 9-3-1	Rands 9-2-1	Rands 9-3-1	Randnr 9-2-1	Randnr 9-3-1	Randnc 9-2-1	Randnc 9-3-1	Rand 9-2-1	Rand 9-3-1	NW 9-2-1	NW 9-3-1	Min. % Error
4.171	4.107	3.126	4.416	3.126	4.41	**-0.98**	-4.29	4.03	4.53	4.57	5.15	3.92	3.80	0.98
4.416	4.243	4.546	3.668	4.546	3.66	-4.61	-15.1	**2.80**	7.05	7.31	6.53	5.80	4.60	2.80
3.842	**3.515**	6.368	4.002	6.368	4.00	-10.5	-30.7	4.68	9.889	11.99	10.54	8.846	5.51	3.51
0.989	**0.614**	5.646	2.384	5.646	2.38	-16.4	-41.6	4.84	10.77	13.90	16.14	9.311	4.50	0.61
-0.55	-0.75	4.754	1.220	4.754	1.22	-15.2	-36.1	5.62	8.545	11.46	14.82	8.899	3.93	0.55
-3.93	-4.02	-0.82	-2.98	-0.82	-2.9	-12.0	-23.6	**0.05**	2.311	2.363	4.191	1.475	-1.40	0.05
-1.85	-2.57	-0.69	-0.35	-0.69	-0.3	-12.9	-23.1	-0.59	-1.17	-1.76	**-0.25**	-2.31	-2.24	0.25
-2.00	-2.63	-3.72	-3.03	-3.72	-3.0	-18.5	-32.3	-3.79	-4.71	-5.62	-4.00	-6.60	-4.04	2.00
1.208	1.511	-2.50	-1.88	-2.50	-1.8	-20.2	-34.9	-1.92	-2.58	-3.21	**-1.17**	-4.45	-1.37	1.17
-3.44	**-2.34**	-6.85	-5.32	-6.85	-5.3	-29.6	-49.4	-6.69	-7.92	-8.94	-6.18	-10.3	-6.66	2.34
-0.17	0.630	-3.50	-4.45	-3.50	-4.4	-32.1	-60.6	-4.10	-5.70	-7.42	-3.78	-8.77	-4.88	0.17
0.270	1.182	-12.1	-4.55	-12.1	-4.5	-33.0	-53.2	-7.85	-7.41	-9.82	-12.7	-11.8	-5.44	0.27
5.886	6.720	**0.189**	2.279	**0.189**	2.27	-9.58	-20.4	1.77	2.923	3.052	0.439	1.300	2.75	0.18
2.471	3.608	3.308	**1.162**	3.308	**1.16**	-13.4	-32.6	1.53	6.802	8.479	5.692	5.554	2.36	1.16
2.920	3.243	7.879	4.327	7.879	4.32	-18.7	-47.3	7.09	13.19	17.58	18.73	12.34	7.24	2.92
2.757	**2.736**	8.930	5.127	8.930	5.12	-16.6	-42.9	8.66	13.11	18.43	22.72	14.08	8.44	2.73
-2.52	-2.72	4.374	**-0.47**	4.374	-0.4	-18.6	-42.1	5.85	9.596	10.63	14.39	8.851	2.67	0.47
-5.38	-5.88	-4.14	-5.59	-4.14	-5.5	-12.3	-21.6	-3.30	-1.19	-1.56	**-0.35**	-2.75	-3.64	0.35
-3.91	-5.93	-2.84	-3.05	-2.84	-3.0	-24.3	-43.6	-2.33	-3.41	-4.34	**-1.67**	-5.51	-3.44	1.67
-8.45	-8.47	-9.74	**-6.34**	-9.74	**-6.3**	-52.2	-90.4	-10.3	-12.5	-14.3	-9.05	-16.1	-11.2	6.34
1.047	1.290	-10.5	-6.26	-10.5	-6.2	-64.6	-110.	-8.89	-12.0	-15.0	-9.78	-18.5	-8.06	1.04

Table 3.1. Percentage Error in Hourly Load Forecasting by Proposed RNN Model

A.*Rand-(A/2) 9-2-1	A.*Rand-(A/2) 9-3-1	2.*Rand-1 9-2-1	2.*Rand-1 9-3-1	Rands 9-2-1	Rands 9-3-1	Randnr 9-2-1	Randnr 9-3-1	Randnc 9-2-1	Randnc 9-3-1	Rand 9-2-1	Rand 9-3-1	NW 9-2-1	NW 9-3-1	
62.22	68.77	106.6	72.92	106.6	72.9	4.37E +02	8.57E +02	96.8	147.3	181.8	168.3	167.7	98.3	**Total Abs. % Error**
6	4	1	3	1	2	1	0	2	0	0	4	0	0	**No. of Min. % Error**
10	9	11	12	11	12	**21**	**21**	10	10	10	10	10	11	**Negative % Error**
12	11	5	9	5	8	1	0	7	5	3	5	4	6	**Error less than 3%**
2.963	3.275	5.079	3.472	5.07	3.47	20.81	40.82	4.61	7.01	8.65	8.01	7.98	4.68	**Avg. hourly % Error**
10.88	11.06	10.5	10.51	10.4	11.1	10.83	**10.41**	11.9	10.7	11.0	11.0	10.4	10.8	**Computation Time**

Table 3.2. Performance Index Comparison of Proposed RNN Model

In Table.3.1, the prediction performance of the proposed RNN is tabulated taking into consideration type of initialization method and number of hidden neurons. The first row gives the formula / standard Matlab functions used to initialize the parameters (weight and bias) of neurons.

The value of A is taken as 0.72, Rand, Randnr, Rands, Randnc are standard Matlab random value generation functions. NW is Nguyen Widrow method of parameter initialization. The network is trained and tested with same set of historical data, so that we can select the parameter initialization method which will give the least Mean Average Percentage Error (MAPE). After the network is trained, it is subjected to testing data for prediction of next 21 (twenty one) hours load. The % prediction errors for each type of initialization method are delineated column wise under the respective initialization methods.

In Table. 3.2., first row signifies summation of absolute percentage errors, second row gives number of minimum percentage errors provided by each method for a given testing data set, third row gives the number of negative % errors, fourth row gives the MAPE and fifth row gives the computation time required for testing for respective parameter initialization method.

3.3. Functional Link Artificial Neural Network

Pao originally proposed FLANN and it is a novel single layer ANN structure capable of forming arbitrarily complex decision regions by generating nonlinear decision boundaries. Here, the initial representation of a pattern is enhanced by using nonlinear function and thus the pattern dimension space is increased. The functional link acts on an element of a pattern or entire pattern itself by generating a set of linearly independent function and then evaluates these functions with the pattern as the argument. Hence separation of the patterns becomes possible in the enhanced space. The use of FLANN not only increases the learning rate but also has less computational complexity. Pao *et al* have investigated the learning and generalization characteristics of a random vector FLANN and compared with those attainable with MLP structure trained with back propagation algorithm by taking few functional approximation problems. A FLANN structure with two inputs is shown in Fig. 3.6.

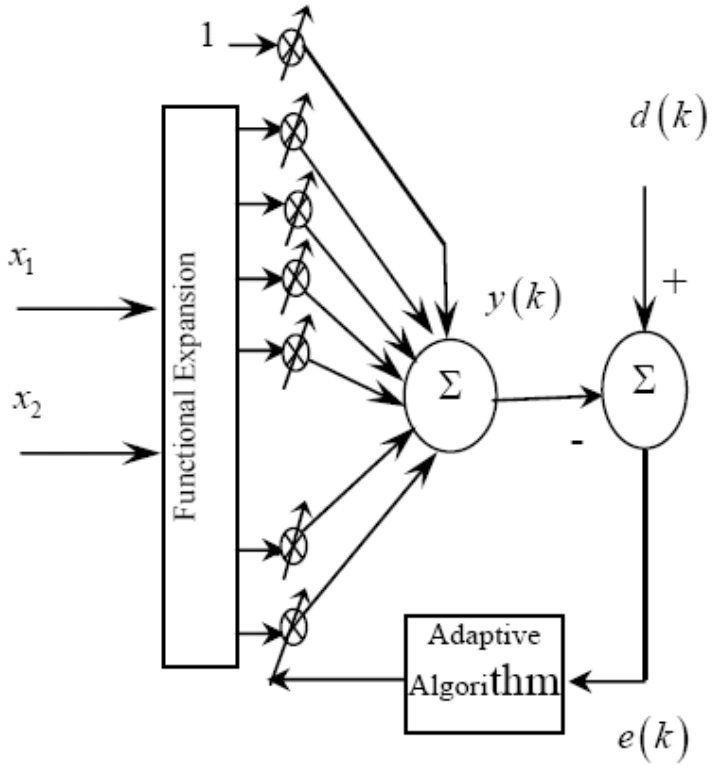


Fig. 3.7. Structure of the FLANN

3.3.1 Introduction

Many Functional Link Artificial Neural Network (FLANN) approach have been proposed, which incorporate the temperature, and historical load for short term load forecasting. In Indian context load variation with respect to temperature variation is almost dependent upon the seasonal change. Seasonal change is always gradual, and an online FLANN is trained every 24 hours to predict next 24 hour load or trained every hour to predict next hour load. So, temperature variations due to seasonal change need not to be taken as an explicit parameter for input. Instead, surge in cooking load in morning hours and commercial, lighting and domestic load in evening hours should be incorporated appropriately as parameters for prediction.

The objective of the present approach is to study FLANN models using Chebyshev, Trigonometric, and Algebraic variants to identify a time-series load model incorporating the non-linearity due to hour of the day, and day of the week etc. In these variants the input pattern is expanded using the respective polynomials. The motive behind usage of FLANN is that it requires very less computational resources for hardware implementation. A FLANN

network consists of a functional expansion block and a single layer perceptron network. The main purpose of the functional expansion block is to increase the dimension of the input pattern so as to enhance its representation in a high dimensional space.

3.3.2 Chebyshev Expansion

In Chebyshev variant of FLANN, the input vector under goes Chebyshev polynomial functional expansion to form the high dimensional space [58,59]. These higher order Chebyshev polynomials for -1<x<1, using the recursive formula given as:

$$T_{n+1} = 2_x T_n(X) - T_{n-1}(X)$$

Chebyshev polynomial is computationally more efficient than trigonometric or algebraic polynomial expansion.

The parameters of the FLANN model with Chebyshev expansion for next hour load forecasting are as given below:

- Activation function: tanh or tansig or logsig
- Training algorithm: Back-Propagation
- Learning rate (α): values as shown in table 3.3.
- No. of input variables: 9 (same day: hr-1, hr-2, hr-3; day-1: hr, hr-1, hr-2; day-7: hr, hr-1, hr-7)
- No of output variable: 1 (same day: hr)
- No of data sets in each Epoch: 63
- No. of Epochs for training: 150
- Functional expansion: $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), x = input$

The Chebyshev FLANN model was tested with many combinations of 'α' value and activation functions. The MAPE thus obtained are discussed in Table 3.3. The forecasted load pattern is shown in dashed line and actual load pattern in dark line. Other 'α' values and activation functions did not significantly impact the load forecast accuracy.

The minimum Mean Average Percentage Error (MAPE) was found to be 3.19 % by a network with $\alpha = 0.003$ and tansig activation function (this is shown in Fig. 3.8.(e)).

3.3.3 Simulation Results

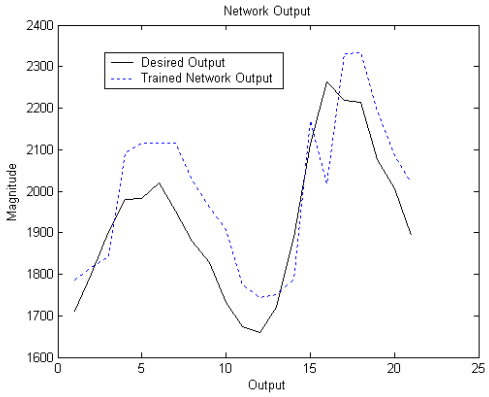


Fig. 3.8.(a) Logsig activation function, α = .003

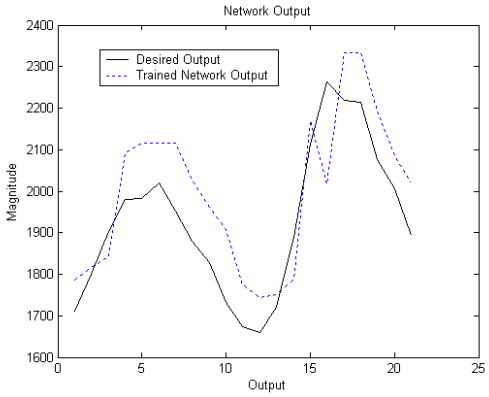


Fig. 3.8.(b) Logsig activation function, α = .005

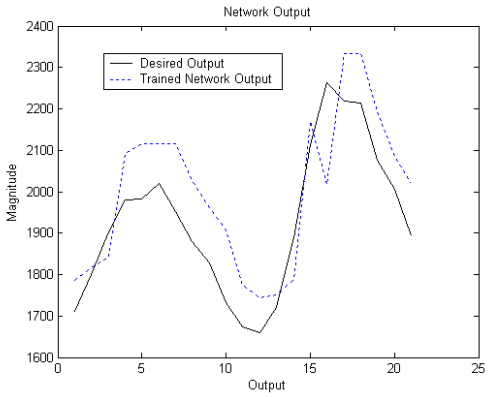


Fig. 3.8.(c) Logsig activation function, α = .007

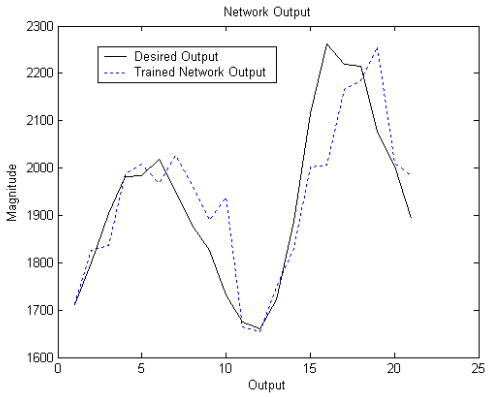


Fig. 3.8.(d) Tansig activation function, α = .001

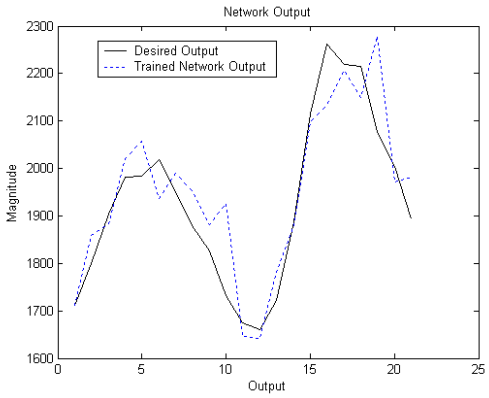


Fig. 3.8.(e) Tansig activation function, α = .003

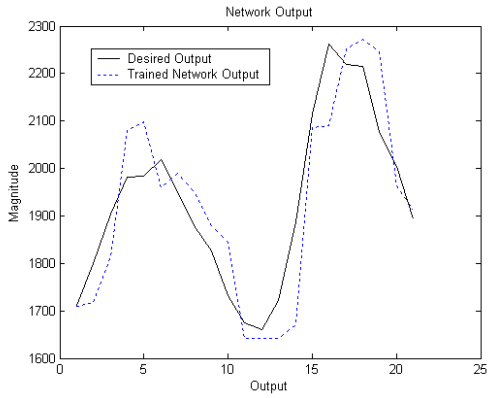


Fig. 3.8.(f) Tansig activation function, α = .005

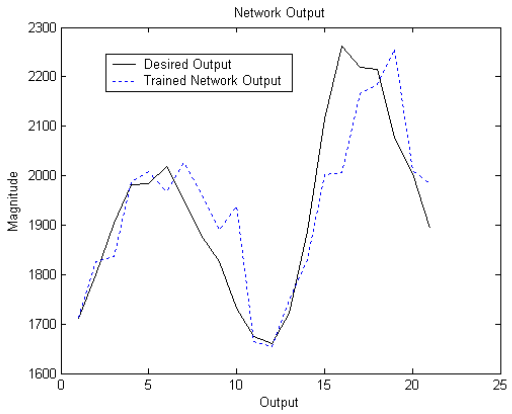


Fig. 3.8.(g) Tanh activation function, α = .001

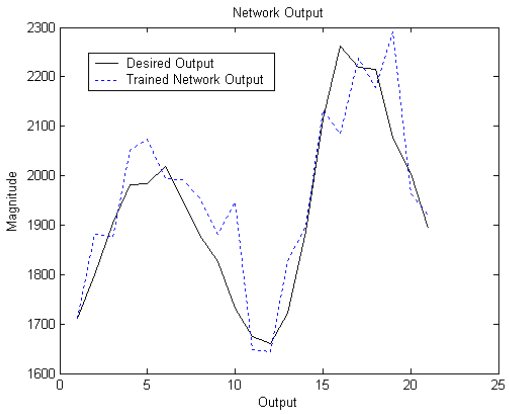


Fig. 3.8.(h) Tanh activation function, α = .003

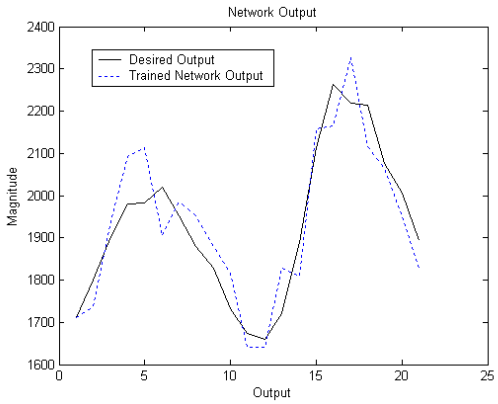


Fig. 3.8.(i) Tanh activation function, α = .005

Fig. 3.8. Hour Ahead Prediction by Chebyshev FLANN with Different Activation Function and 'α' Value

Logsig .001	Logsig .003	Logsig .005	Tansig .001	Tansig .003	Tansig .005	Tanh .001	Tanh .003	Tanh .005	
117.57	117.80	117.80	72.81	**66.96**	81.87	72.81	71.20	72.70	Total Abs. % Error
1	1	1	**7**	4	3	**7**	2	6	No. of Min. % Error
18	**18**	**18**	12	10	10	12	13	10	Negative % Error
3	3	3	11	11	11	11	**12**	9	Error less than 3%
5.60	5.61	5.61	3.47	**3.19**	3.90	3.47	3.39	3.46	MAPE

Table 3.3. Performance Index Comparison for Hourly Load Forecasting by Chebyshev FLANN

3.3.4 Trigonometric Expansion

In Trigonometric variant of FLANN, the input vector under goes Trigonometric polynomial functional expansion to form the high dimensional space. Let there be an input vector

$$x(k) = \{x(1), x(2),\ldots.x(m)\}$$

Each element in the vector in x(k) is expanded into several terms. The main purpose of the functional expansion block is to increase the dimension of the input pattern so as to enhance its representation in a high-dimensional space. The expanded input vector, X is obtained by using trigonometric functions given by

$$X = [\, x_1\cos(\Pi x_1)\ \sin(\Pi x_1)\ \ldots\ x_2\cos(2\Pi x_2)\ \sin(2\Pi x_2)\ \ldots.\ x_m\cos(m\Pi x_m)\ \sin(m\Pi x_m)]$$

The parameters of the FLANN model with Trigonometric expansion for next hour load forecasting are as given below:

- Activation function: tansig or logsig or tanh
- Training algorithm: Back-Propagation
- Learning rate (α): values as shown in table 3.4.
- No. of input variables: 9 (same day: hr-1, hr-2, hr-3; day-1: hr, hr-1, hr-2; day-7: hr, hr-1, hr-7)
- No of output variable: 1 (same day: hr)
- No of data sets in each Epoch: 63
- No. of Epochs for training: 150
- Functional expansion: $1, \sin(x),\ldots.\sin(5x), \cos(x)\ldots.\cos(5x), x = input$

The Trigonometric FLANN model was tested with many combinations of 'α' value and activation functions. The MAPE thus obtained are discussed in Table 3.4. The forecasted load pattern is shown in dashed line and actual load pattern in dark line. Other 'α' values and activation functions did not significantly impact the load forecast accuracy.

The minimum Mean Average Percentage Error (MAPE) was found to be 3.86 % by a network with $\alpha = 0.001$ and tansig activation function (this is shown in Fig. 3.9.(a)).

3.3.5 Simulation Results

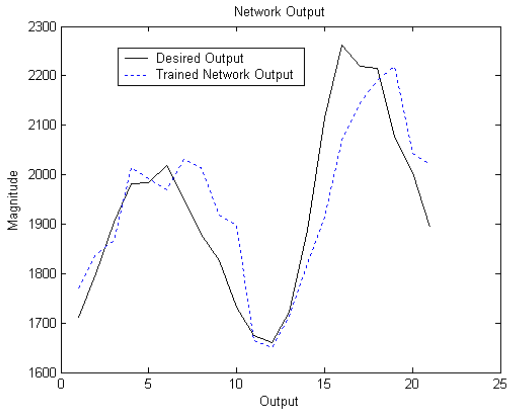


Fig. 3.9.(a) Tansig activation function, α = .001

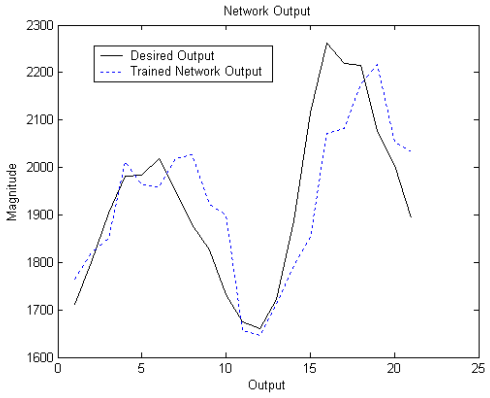


Fig. 3.9.(b) Tansig activation function, α = .003

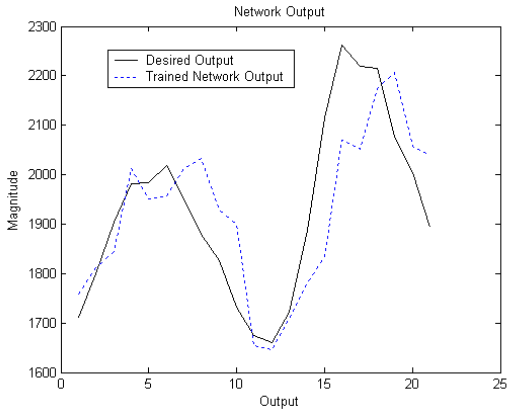


Fig. 3.9.(c) Tansig activation function, α = .005

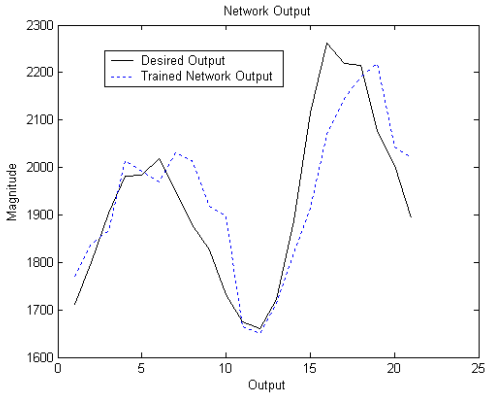


Fig. 3.9.(d) Tanh activation function, α = .001

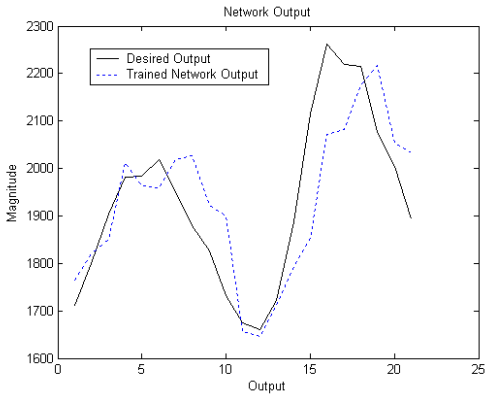


Fig. 3.9.(e) Tanh activation function, α = .003

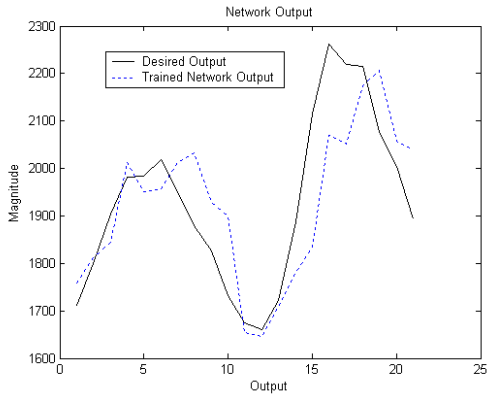


Fig. 3.9.(f) Tanh activation function, α = .005

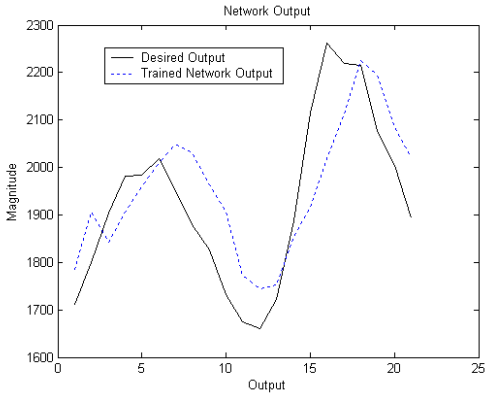


Fig. 3.9.(g) Logsig activation function, α = .001

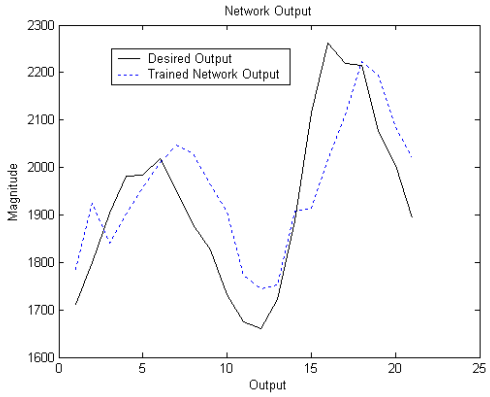


Fig. 3.9.(h) Logsig activation function, α = .003

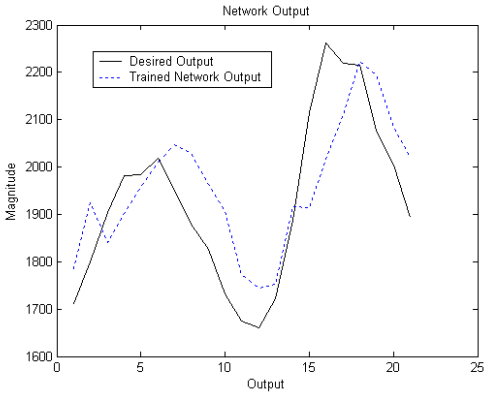


Fig. 3.8.(i) Logsig activation function, α = .005

Fig. 3.9. Hour Ahead Prediction by Trigonometric FLANN with Different Activation Function and 'α' Value

Logsig .001	Logsig .003	Logsig .005	Tansig .001	Tansig .003	Tansig .005	Tanh .001	Tanh .003	Tanh .005	
105.88	106.20	106.71	**80.96**	91.37	95.28	80.96	91.37	95.28	Total Abs. % Error
2	3	2	**11**	1	2	**11**	2	3	No. of Min. % Error
13	**14**	**14**	11	10	10	11	10	10	Negative % Error
13	**14**	**14**	11	10	10	11	10	10	Error less than 3%
5.04	5.06	5.08	**3.86**	4.35	4.54	3.86	4.35	4.54	MAPE

Table 3.4. Performance Index Comparison for Hourly Load Forecasting by Trigonometric FLANN

3.3.6 Algebraic Expansion

In Algebraic variant of FLANN, the input vector under goes Algebraic polynomial functional expansion to form the high dimensional space. Let there be an input vector

$$x(k) = \{x(1), x(2),\ldots x(m)\}$$

Each element in the vector in x(k) is expanded into several terms. The main purpose of the functional expansion block is to increase the dimension of the input pattern so as to enhance its representation in a high-dimensional space. The expanded input vector, X is obtained by using trigonometric functions given by

$$X = [\ 1\ x(1)\ x(1)^2\ x(1)^3\ \cdots\cdots 1\ x(2)\ x(2)^2\ \ldots\ldots\ldots 1\ x(m)\ x(m)^2\ x(m)^3\cdots\cdots]$$

The parameters of the FLANN model with Algebraic expansion for next hour load forecasting are as given below:

- Activation function: tansig or logsig or tanh
- Training algorithm: Back-Propagation
- Learning rate (α): values as shown in table 3.5.
- No. of input variables: 9 (same day: hr-1, hr-2, hr-3; day-1: hr, hr-1, hr-2; day-7: hr, hr-1, hr-7)
- No of output variable: 1 (same day: hr)
- No of data sets in each Epoch: 63
- No. of Epochs for training: 150
- Functional expansion: $1, x^2, x^3, \ldots\ldots\ldots\ldots x^{10}$

The Algebraic FLANN model was tested with many combinations of 'α' value and activation functions. The MAPE thus obtained are discussed in Table 3.5. The forecasted load pattern is shown in dashed line and actual load pattern in dark line. Other 'α' values and activation functions did not significantly impact the load forecast accuracy.

The minimum Mean Average Percentage Error (MAPE) was found to be 3.1512 % by a network with $\alpha = 0.09$ and tansig activation function (this is shown in Fig. 3.10.(c)).

3.3.7 Simulation Results

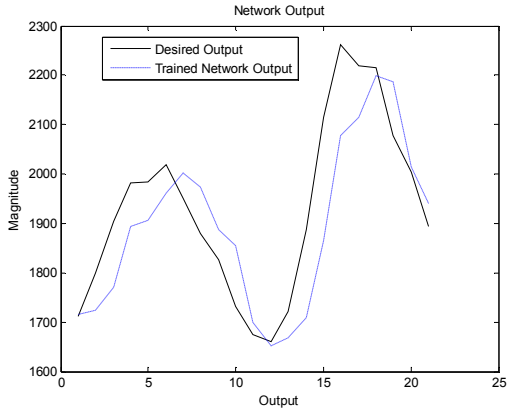


Fig. 3.10.(a) Tansig activation function, α = .01

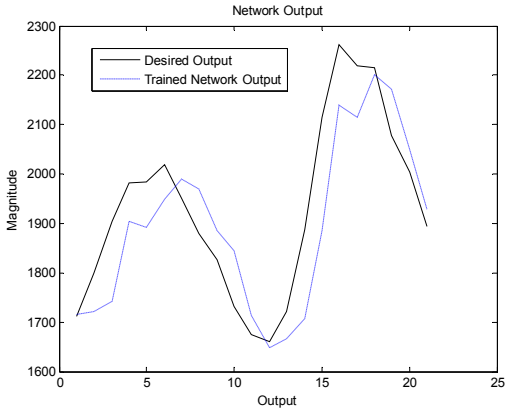


Fig. 3.10.(b) Tansig activation function, α = .03

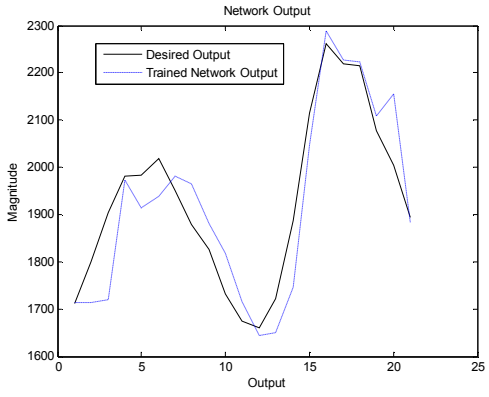


Fig. 3.10.(c) Tansig activation function, α = .09

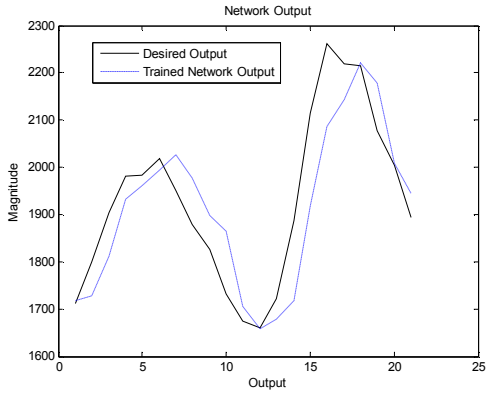


Fig. 3.10.(d) Tanh activation function, α = .001

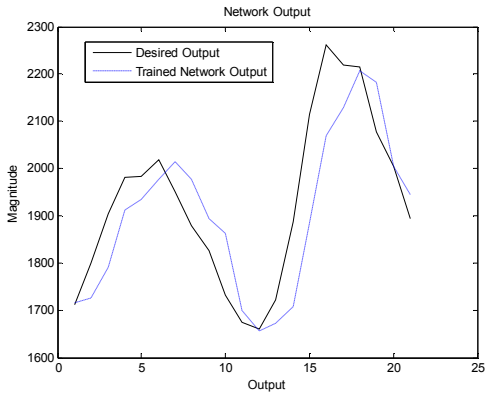


Fig. 3.10.(e) Tanh activation function, α = .003

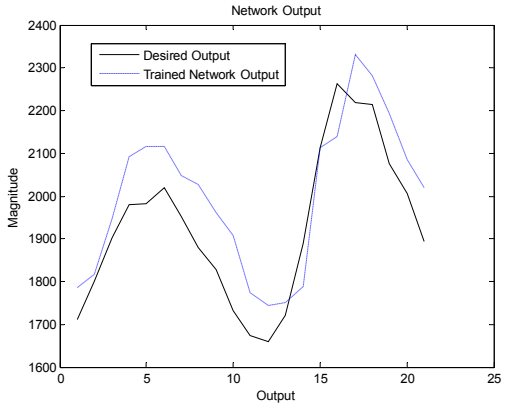


Fig. 3.10.(f) Logsig activation function, α = .003

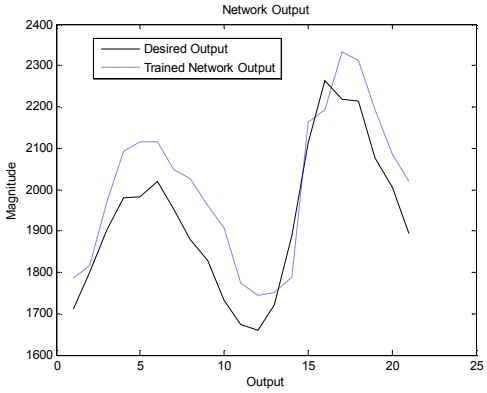


Fig. 3.10.(g) Logsig activation function, α = .005

Fig. 3.8. Hour Ahead Prediction by Algebraic FLANN with Different Activation Function and 'α' Value

Logsig .003	Logsig .005	Tanh .001	Tanh .003	Tansig .01	Tansig .03	Tansig .09	Present Page
102.8874	105.492	76.2299	83.4464	88.7239	87.4709	**66.1749**	Total Abs. % Error
3	2	4	0	0	0	3	No. of Min. % Error
18	**19**	10	8	9	9	11	Negative % Error
4	3	10	9	8	7	11	Error less than 3%
4.8994	5.02342857	3.62999524	3.9736381	4.2249	4.1653	**3.1512**	Avg. hourly % Error

Table 3.5. Performance Index Comparison for Hourly Load Forecasting by Algebraic FLANN

Chapter 4

SHORT TERM LOAD FORECASTING USING EVOLUTIONARY ALGORITHMS

4. STLF Using Evolutionary Algorithm

4.1. MLPNN Trained by Genetic Algorithm

4.1.1 Introduction

Genetic Algorithm (GA) is a directed random search technique [60] that is widely applied in optimization problems [60-62]. This is especially useful for complex optimization problems where the number of parameters is large and the analytical solutions are difficult to obtain. GA can help to find out the optimal solution globally over a domain. This technique has been applied in different areas such as fuzzy control, path planning [63], modeling and classification [64] etc.

There are two kinds of genetic operators, namely crossover and mutation. For crossover mechanisms, two-point cross over, multipoint crossover, arithmetic crossover, and heuristic crossover have been reported [60, 65-67]. For mutation mechanisms, boundary mutation, uniform mutation, and nonuniform mutation can be found [60, 65-67]. Three steps are used to generate offspring: copying the parents, determining the mutations to be performed, and mutating the copy. In our work reported herein, GA is based on Steady-State Algorithm with Fitness – Proportional Selection procedure. The population is composed of 10 bit binary number chromosomes. The algorithm is described in Fig.4.1. in a self explanatory manner.

The parameters of the neural system for hour ahead load forecasting are as given below:

MLP Neural Network Parameters

- No. of layers: 3 (Input layer, Hidden layer, Output layer)
- No of neurons in hidden layer: 4
- No of neurons in output layer: 1
- Activation function of hidden layer: logsig or tansig or tanh
- Activation function of output layer: Linear
- Training algorithm: Genetic Algorithm
- No. of input variables: 9 (same day: hr-1, hr-2, hr-3; day-1: hr, hr-1, hr-2; day-7: hr, hr-1, hr-7)

- No of output variable: 1 (same day: hr)
- No of data sets in each generation: 63

```
initialize P                        // P: initial population
iter                                // iter: number of iteration
    begin
        batch                       //batch: total no. of inputs
                begin
                        calculate the MLP output and thus error
                end

        calculate MSE and thus fitness function f(P(iter))
        P(iter) is sorted in decreasing fitness order of f(P(iter))
        iter = iter + 1
        select some pairs of parents p1 and p2 from P(iter-1) with higher fitness value in the fitness
        order
         perform genetic operations (crossover and mutation) to produce p1' and p2'
          replace p1 and p2 with p1' and p2' and thus reproducing a new generation P(iter)
    end
    best individual from population P is chosen as the output
```

Fig. 4.1. Procedure for Genetic Algorithm

Genetic Algorithm Parameters

- Population size: 60
- Crossover probability: 80 percent
- Mutation probability: 20 percent
- Selection: fitness-proportional
- Number of generations: 150
- Chromosome: 10 bit binary number
- Fitness function: 1 / (1+MSE2)
- No of cost function (s): 1, depending on the final output dependent error

Logsig, tansig, and tanh activation functions were taken for testing purpose. After much experimentation the MLP-NN and GA parameters were selected as above. The MAPE obtained by using logsig, tansig and tanh activation functions were found to be 3.3909%, 4.7513% and 3.7416% respectively, as described in Table. 4.1. The corresponding results are

shown in Fig. 4.2. The forecasted load pattern is shown in dashed line and actual load pattern in solid line. Other activation functions did not significantly impact the load forecast accuracy.

The minimum Mean Average Percentage Error (MAPE) was found to be 3.3909 % by a network with logsig activation function (as shown in Fig. 4.2.(a)).

4.1.2 Simulation Results

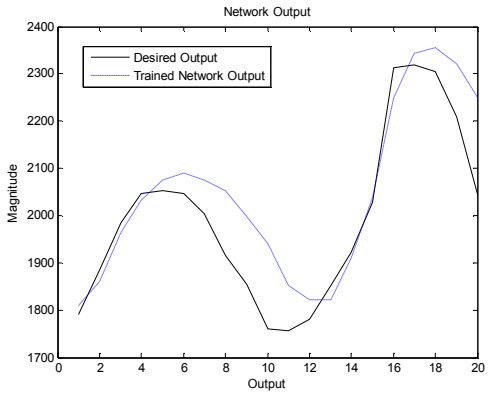


Fig. 4.2.(a) Logsig activation function

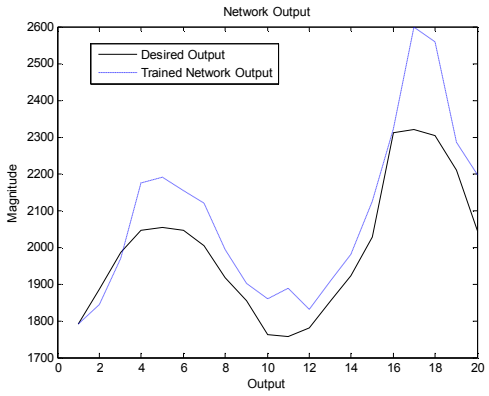


Fig. 4.2.(b) Tansig activation function

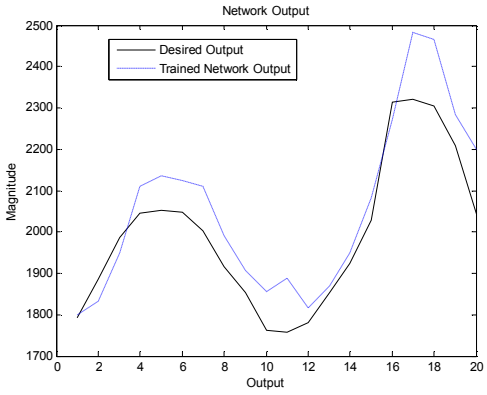


Fig. 4.2.(c) Tanh activation function,

Fig. 4.2. Hour Ahead Prediction by MLPNN trained by GA with Different Activation Functions and Single Fitness f^n [e = sum(error)]

tansig	logsig	tanh	
89.3791	86.109	74.3246	Total Abs. % Error
14	4	2	No. of Min. % Error
11	9	10	Negative % Error
11	5	7	Error less than 3%
3.1934	4.3054	3.7162	Avg. hourly % Error

Table 4.1. Performance Index Comparison for Hourly Load Forecasting by MLPNN-GA

4.1.3 Proposed Hybrid Genetic Algorithm Approach

In conventional neural network structures Genetic Algorithm is used to minimize the final error but in this proposed method the error contributed by each neuron is calculated through back propagation from the Jacobian matrix. So, instead of having one fitness function for all the genes, we have as many number of fitness functions as number of neurons.

The parameters of the system for next hour load forecasting are as given below:

MLP Neural Network Parameters

- No. of layers: 3 (Input layer, Hidden layer, Output layer)
- No of neurons in hidden layer: 4
- No of neurons in output layer: 1
- Activation function of hidden layer: logsig or tansig or tanh
- Activation function of output layer: Linear
- Training algorithm: Genetic Algorithm + Back-Propagation
- No. of input variables: 9 (same day: hr-1, hr-2, hr-3; day-1: hr, hr-1, hr-2; day-7: hr, hr-1, hr-7)
- No of output variable: 1 (same day: hr)
- No of data sets in each generation: 63

Hybrid GA Parameters

- Population size: 60
- Crossover probability: 80 percent
- Mutation probability: 20 percent
- Selection: fitness-proportional

- Number of generations: 150
- Chromosome: 10 bit binary number
- Fitness function: 1 / (1+MSE2)
- No of cost function (s): 5, depending on the final output dependent error and error contributed by each hidden neuron. The error contribution of each hidden neuron is calculated by back propagating the final error.

Logsig, tansig, and tanh activation functions were taken for testing purpose. After much experimentation the MLP-NN and Hybrid-GA parameters were selected as above. The MAPE obtained by using logsig, tansig and tanh activation functions were found to be 4.3054%, 3.1934% and 3.7162% respectively, as described in Table. 4.2. The corresponding results are shown in Fig. 4.3.(a),4.3.(c), and 4.3.(e). In Fig. 4.3.(b), 4.3.(d) and 4.3.(f) the graph shows the absolute mean square error for the given number of epochs. The forecasted load pattern is shown in dashed line and actual load pattern in solid line. Other activation functions did not significantly impact the load forecast accuracy.

The minimum Mean Average Percentage Error (MAPE) was found to be 3.1934 % by a network with Tansig activation function (as shown in Fig. 4.3.(c)).

4.1.4 Simulation Results

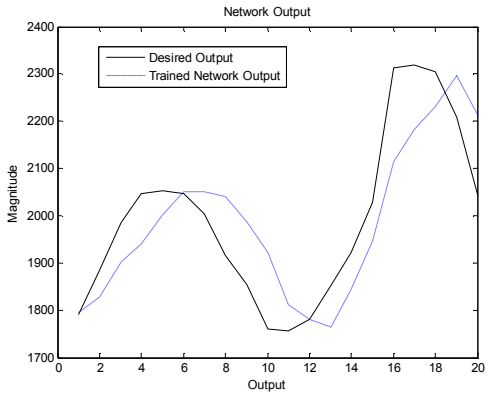


Fig. 4.3.(a) Logsig activation function

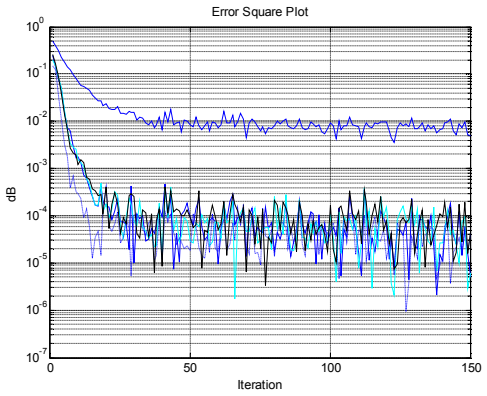


Fig. 4.3.(b) Logsig activation function

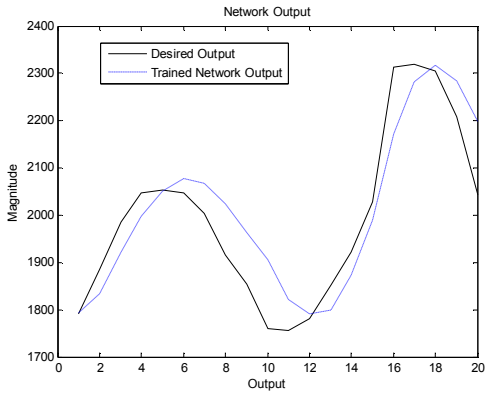


Fig. 4.3.(c) Tansig activation function

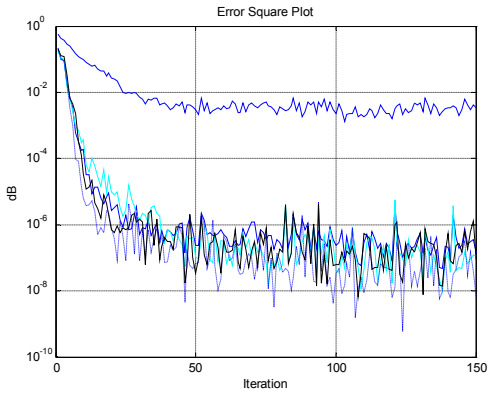


Fig. 4.3.(d), Tansig activation function

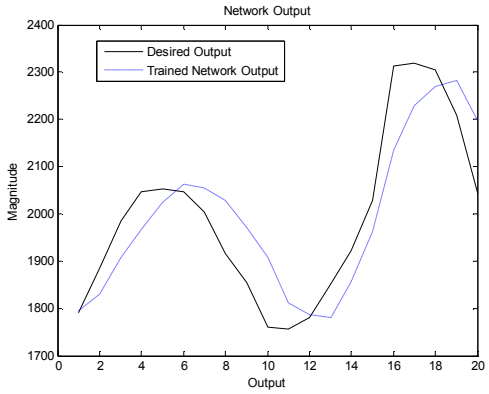


Fig. 4.3.(e) Tanh activation function

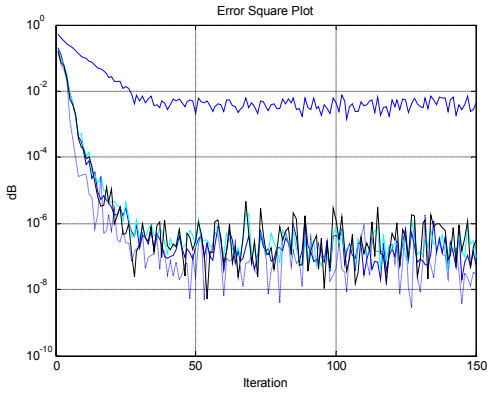


Fig. 4.3.(f) Tanh activation function

Fig. 4.3. Hour Ahead Prediction by MLPNN trained by GA-BP Hybridized Algorithm with Different Activation Functions and Individual Fitness fⁿ [e = sum(error)]

tansig	logsig	tanh	
95.0255	67.8183	74.8329	Total Abs. % Error
5	10	5	No. of Min. % Error
17	14	17	Negative % Error
8	13	9	Error less than 3%
4.7513	3.3909	3.7416	Avg. hourly % Error

Table 4.2. Performance Index Comparison for Hourly Load Forecasting by MLPNN-(BP –GA Hybrid)

4.2. MLPNN Trained by Particle Swarm Optimization

4.2.1 Introduction

Particle Swarm Optimization (PSO) is an idea based on human social behavior. Kennedy and Eberhert [68] in 1994 presented the concept. It models problem as a set of *n* particles each representing a dimension of solution space. These particles move in solution space in search of optimal solution. The particles follow three principles i.e. evaluating: learning through self experience, Comparing: learning through comparative study and Imitating: learning through adapting the best trend.

PSO has many similarities with genetic algorithm (GA). But it does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity. They also have memory, which is important to the algorithm. One of the advantages of PSO is that it takes real numbers as particles unlike GA, which needs to change to binary encoding or special genetic operators have to be used. In GA, chromosomes share information with each other. So, the whole population moves like one group towards optimal area. In PSO, only global best or local best gives information to others. It is a one way information sharing mechanism. Compared with GA, all the particles tend to converge to the best solution quickly even in the local version in most cases. In PSO, we have got a position parameter, a velocity parameter to be updated for all the dimensions of all the particles. The updation logic lies with the global best parameter of all the particles and the local best parameter of a single particle. Our PSO is based on continuous state variables with real values. The velocity and position updation is done as per the following equations:

$$\mathbf{V}_i^{k+1} = \mathbf{w} * \mathbf{V}_i^{k} + \boldsymbol{c}\mathbf{1} * \text{rand}_1 * (\mathbf{pbest}_i - \mathbf{X}_i^{k}) + \text{c2} * \text{rand}_2 * (\mathbf{gbest} - \mathbf{X}_i^{k})$$

$$\mathbf{X}_i^{k+1} = \mathbf{X}_i^{k} + \mathbf{V}_i^{k+1}$$

Details are explained in Fig. 4.4. in a self explanatory way.

initialize **S** with random **V** and **X** // **S**: initial swarm, **V**: velocity, **X**: position

initialize w, c1, c2 // w: weighting function, c1 = c2: weighting factors

k → // ***k***: number of iteration

begin

$\mathbf{X}_i^k$: position of a particle

$\boldsymbol{pbest}_i$: personal best position of i^{th} particle

$\mathbf{V}_i^k$: velocity of i^{th} particle

gbest: global best of all the particles of **S**

batch → //**batch**: total no. of inputs

begin

calculate the MLP output and thus error

end

calculate MSE for each particle in swarm

For each particle ***i*** in swarm

If ($\mathbf{X}_i^k$ < $\mathbf{pbest}_i$)

Then $\mathbf{pbest}_i = \mathbf{X}_i^k$

If ($\mathbf{pbest}_i$ < **gbest**)

Then **gbest** = $\mathbf{pbest}_i$

Position and velocity are then updated as per the following equations:

$$\mathbf{V}_i^{k+1} = \mathbf{w} * \mathbf{V}_i^k + \boldsymbol{c}\mathbf{1}* \text{rand}_1 * (\mathbf{pbest}_i - \mathbf{X}_i^k) + \text{c2}* \text{rand}_2 * (\mathbf{gbest} - \mathbf{X}_i^k)$$

$$\mathbf{X}_i^{k+1} = \mathbf{X}_i^k + \mathbf{V}_i^{k+1}$$

end

best individual from swarm **S** is chosen as the output

Fig. 4.4. Procedure for Particle Swarm Optimization

The parameters of the system for next hour load forecasting are as given below:

MLP Neural Network Parameters

- No. of layers: 3 (Input layer, Hidden layer, Output layer)
- No of neurons in hidden layer: 4
- No of neurons in output layer: 1
- Activation function of hidden layer: logsig or tansig or tanh
- Activation function of output layer: Linear
- Training algorithm: Particle Swarm Optimization Algorithm + Back-Propagation
- No. of input variables: 9 (same day: hr-1, hr-2, hr-3; day-1: hr, hr-1, hr-2; day-7: hr, hr-1, hr-7)
- No of output variable: 1 (same day: hr)
- No of data sets in each generation: 21

Particle Swarm Optimization Parameters

- Swarm size: 10
- Particle dimension: 45
- Initial weight (w1): 0.9
- Final weight (w2): 0.4
- Constriction factor: 0.7298
- Weighting factor (c1 = c2): 1.4962
- Particle position range: 0 – 0.1
- Particle velocity range: 0 – 0.001
- Number of iterations (kmax): 100
- Weighting function: w1-((w1-w2)/ kmax) * k

Logsig, tansig, and tanh activation functions were taken for testing purpose. After much experimentation the MLP-NN and PSO parameters were selected as above. The MAPE obtained by using logsig, tansig and tanh activation functions were found to be 4.2118%, 4.469% and 4.469% respectively, as described in Table. 4.3. The corresponding results are shown in Fig. 4.5. The forecasted load pattern is shown in dashed line and actual load pattern in solid line. Other activation functions did not significantly impact the load forecast accuracy.

The minimum Mean Average Percentage Error (MAPE) was found to be 4.2118 % by a network with logsig activation function (as shown in Fig. 4.5.(a)).

4.2.2 Simulation Results

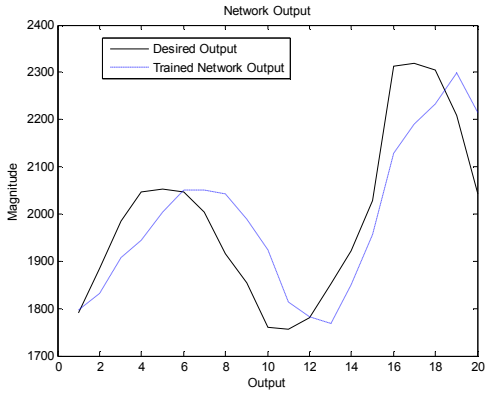


Fig. 4.5.(a) Logsig activation function

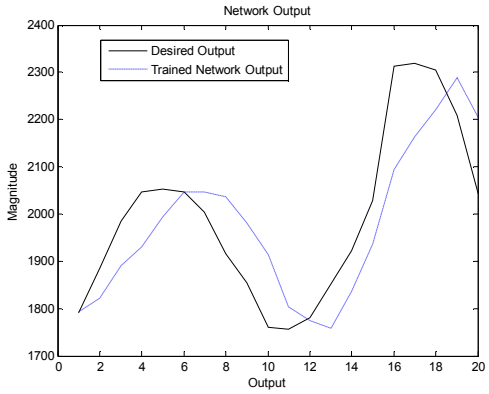


Fig. 4.5.(b) Tansig activation function

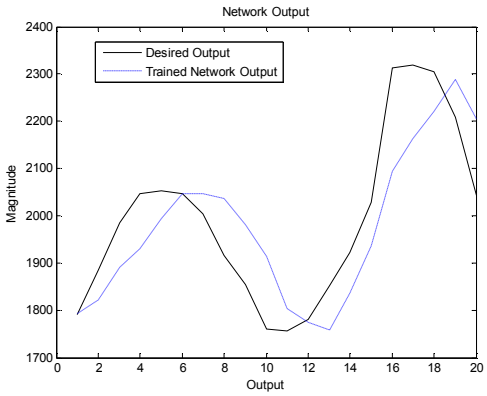


Fig. 4.5.(c) Tanh activation function

Fig. 4.5. Hour Ahead Prediction by MLPNN trained by PSO with Different Activation Functions

tansig	logsig	tanh	
89.3791	84.3767	89.3791	Total Abs. % Error
9	12	9	No. of Min. % Error
8	10	8	Negative % Error
6	5	6	Error less than 3%
4.469	4.2118	4.469	Avg. hourly % Error

Table 4.3. Performance Index Comparison for Hourly Load Forecasting by MLPNN-PSO

4.3. MLPNN Trained by Artificial Immune System

4.3.1 Introduction

Feed forward neural net structures like multi layer perceptron, functional link, wavelet, recurrent or feedback structures like Hopfield, Elman, Multi Feedback [69] and hybrid structures using fuzzy neural networks have been widely proposed for non-stationary forecasting applications and have seen to provide very high degree of predictive accuracy. But many of those papers have been tested on Macky-Glass series or some other smooth differentiable functions rather than actual load data. Actual load data putforths many challenges to design a predictive neural net structure Prominent of those challenges are, data pre-processing, input parameter selection, type of neural net structure selection, computational complexity and training algorithm. Computational complexity is dependent on the structural complexity and training algorithm. This factor becomes important for real time implementation of algorithms in power generation and transmission equipment.

Moreover, evolutionary and behavioral random search algorithms such as genetic algorithm (GA) [70-72], particle swarm optimization (PSO) [73, 74], etc. have previously been implemented for different problems. Infact, genetic algorithms, based on the theory of genetic evolution, due to their parallel search techniques, have attracted much attention in the past, and were successfully implemented to a variety of electrical engineering problems. GA has been deployed to forecast short term load with various modifications over the years. In spite of its successful implementation, GA does posses some weaknesses leading to longer computation time and less guaranteed convergence, particularly in case of epistatic objective function containing highly correlated parameters [75, 76]. Moreover, premature convergence of GA is accompanied by a very high probability of entrapment into the local optimum. In order to alleviate the aforementioned difficulties, this paper proposes a new approach, to forecast short term load, inspired by the characteristics of immune system. Immune system (IS) is a very intricate biological system which accounts for resistance of a living body against harmful foreign entities. Artificial immune system (AIS) imitates the immunological ideas to develop some techniques used in various areas of research [77]. It works on the principle of pattern recognition (distinguishing antibody and antigen) and clonal selection principle, whereby clonal selection algorithm (invariably called as AIS) is implemented to accomplish learning and memory acquisition tasks. In IS, receptors present on the antibodies

are responsible for antibody–antigen interaction. In these interactions, different antibodies have different affinity towards an antigen and the binding strength is directly proportional to this affinity [78]. AIS effectively exploit these interactions and the corresponding affinity by suitably mapping it to fitness (objective function) evaluation. These ideas are further emulated and thereby harnessed into earning, memory and associative retrieval to solve the prediction problems.

In conventional AIS, the fitness function is calculated on basis of the final network error, which is used to update the weights and biases. In back propagation algorithm error contribution of all the neurons is taken into consideration for updation of corresponding weights and biases of neurons. This paper proposes a hybrid Artificial Immune System (AIS) algorithm for predicting short term load. This method has a better convergence, and accuracy than conventional AIS, GA, PSO and ANFIS.

Following this introduction the remaining paper is organized as under. Section 2 provides overview of artificial immune system, while Section 3 analyzes the proposed Hybrid AIS algorithm. Section 4 highlights the system model for load forecast. The experimental results are presented in Section 5 and Section 6 provides the concluding remarks.

4.3.2 Artificial Immune System Approach

In this section, some concepts and technical terms necessary for the development of our model is introduced [79].Generally speaking the main function of the immune system is to limit the damage to the host organism exposed to foreign harmful substances (e.g. bacteria and viruses). These harmful substances are identified by molecules called antigens. The antigens are responsible for triggering an immune response, which consists of secretion of antibodies by B-cells to participate in the recognition and destruction of theses invading antigens. The B-cells are monospecific, that is they have a single type of receptor, thus, no distinction between he B-cell and its receptor is considered in this work. These biological principles of clone generation, proliferation and maturation are modeled into an algorithm termed the clonal selection algorithm (CLONALG).

In [80] de Castro and Von Zuben focused on the clonal selection principle and affinity maturation process of the adaptive immune response in order to develop an algorithm suitable to perform tasks such as machine learning, pattern recognition, and optimization. Their algorithm was evaluated in a simple binary character recognition problem, multimodal

optimization tasks and a combinatorial optimization problem; more specifically the traveling salesman problem (TSP). The main immune aspects taken into account to develop the algorithm, named CLONALG, were: selection and cloning of the most stimulated cells proportionally to their antigenic affinity; death of non-stimulated cells; affinity maturation and selection of cells proportionally to their antigenic affinity; and generation and maintenance of diversity. The algorithm CLONALG works as follows:

1. Generate a set of N candidate solutions (antibody repertoire) in a shape-space to be defined by the problem under study.
2. Select n1 highest affinity cells in relation to the antigen set to be recognized or to the function being optimized.
3. Clone (generate identical copies of) these n1 selected cells. The number of copies is proportional to their affinities: the higher the affinity, the larger the clone size (number of offspring).
4. Mutate with high rates (hyper mutation) these n selected cells with a rate inversely proportional to their affinities the higher the affinity, the smaller the mutation rate.
5. Re-select n2 highest affinity mutated clones to compose the new repertoire.
6. Replace some low affinity cells by new ones.
7. Repeat steps 2 to 6 until a given stopping criterion is met.

The authors characterized CLONALG as an evolutionary like algorithm with the main features of population based search guided by the mechanisms of reproduction, genetic variation and selection. It is important to note however, that though CLONALG is a type of evolutionary algorithm, it was developed using inspiration from the immune system. In contrast, the standard evolutionary algorithms were devised inspired by the neo-Darwinian theory of evolution. Thus, in the former case (CLONALG) the evolutionary theory is used to explain how the algorithm behaves, and in the latter case (EAs) the evolutionary theory was used to create the algorithm. There are however, some important differences between CLONALG and a GA for example. CLONALG performs not only affinity proportionate selection, but also affinity proportional mutation, and there is no crossover. Similarity does exist however, in the fact that both algorithms encode the individuals of the population. When compared with the evolution strategies, for example, again, differences exist between the algorithms. Evolution strategies work with real-valued encoding, while CLONALG works with binary representation, and the affinity proportional mutation in CLONALG is not

controlled by Gaussian distributions. Therefore, no matter which type of evolutionary algorithm is compared with CLONALG, there are always enough differences between them, in terms of inspiration and computation that justify the proposal of CLONALG as an evolutionary algorithm inspired by immunology.

The parameters of the system for next hour load forecasting are as given below:

MLP Neural Network Parameters

- No. of layers: 3 (Input layer, Hidden layer, Output layer)
- No of neurons in hidden layer: 4
- No of neurons in output layer: 1
- Activation function of hidden layer: logsig or tansig or tanh
- Activation function of output layer: Linear
- Training algorithm: Artificial Immune System
- No. of input variables: 9 (same day: hr-1, hr-2, hr-3; day-1: hr, hr-1, hr-2; day-7: hr, hr-1, hr-7)
- No of output variable: 1 (same day: hr)
- No of data sets in each generation: 21

AIS Parameters

- Population size (N): 75
- Clonal population: $N * \left(fitness^{j}(k) \Big/ \sum_{i=1}^{N} fitness^{j}(i) \right)$
- Mutation probability: $M * \left(\max\left(fitness^{j}\right) - fitness^{j}(k) \Big/ \max\left(fitness^{j}\right) - \min\left(fitness^{j}\right) \right)$
- Number of generations: 20
- Antibody (M): 30 bit binary number
- Fitness function: 1 / (1+MSE2)
- No of cost function (s): 1, depending on the final output dependent error

Logsig, tansig, and tanh activation functions were taken for testing purpose. After much experimentation the MLP-NN and AIS parameters were selected as above. The MAPE obtained by using logsig, tansig and tanh activation functions were found to be 5.2756%,

6.9528% and 5.2662% respectively, as described in Table. 4.4. The corresponding results are shown in Fig. 4.6. The forecasted load pattern is shown in dashed line and actual load pattern in solid line. Other activation functions did not significantly impact the load forecast accuracy.

The minimum Mean Average Percentage Error (MAPE) was found to be 5.2662 % by a network with tanh activation function (as shown in Fig. 4.6.(c)).

4.3.3 Simulation Results

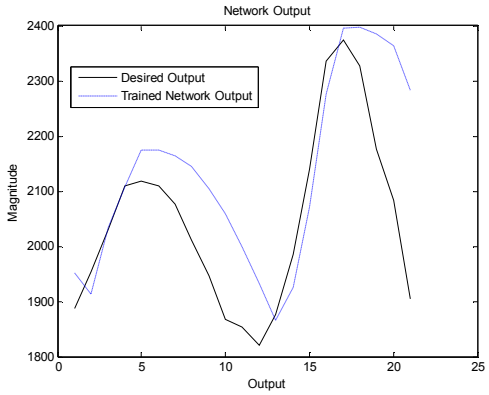


Fig. 4.6.(a) Logsig activation function

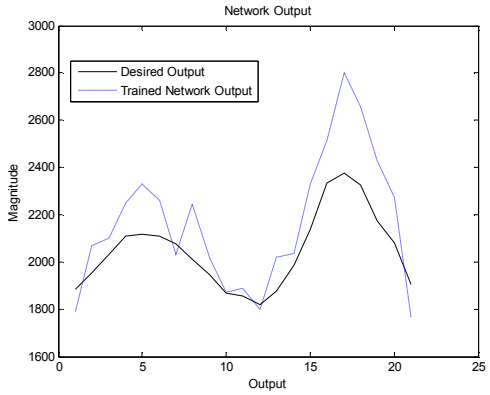


Fig. 4.6.(b) Tansig activation function

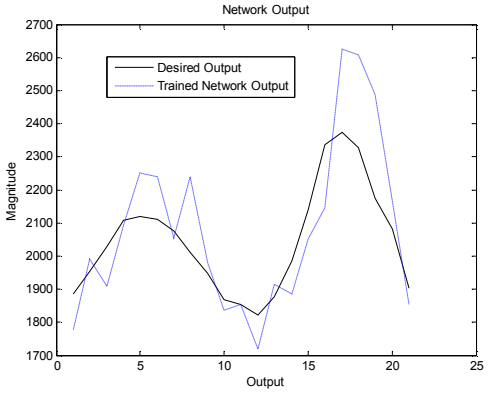


Fig. 4.6.(c) Tanh activation function

Fig. 4.6. Hour Ahead Prediction by MLPNN trained by AIS with Different Activation Functions and Single Fitness fⁿ [e = sum(error)]

tansig	logsig	tanh	
146.0095	110.7885	110.5903	Total Abs. % Error
2	13	6	No. of Min. % Error
16	15	10	Negative % Error
5	8	8	Error less than 3%
6.9528	5.2756	5.2662	Avg. hourly % Error

Table 4.4. Performance Index Comparison for Hourly Load Forecasting by MLPNN-AIS

4.3.4 Proposed Hybrid Immune System Approach

In order to reduce the computational cost, increase convergence speed, and reduce less historical load input, the hybrid AIS algorithm is proposed here. The AIS is implemented to Short term Load Forecasting problem utilizing the following four main features. Firstly, a pool of immune cells or antibodies is generated. This is followed by cloning or copying of the parents. Then, maturation of these clones takes place which is analogous to hyper mutation. Thereafter, the antibody–antigen interaction is evaluated followed by the elimination of self reacting immune cells or lymphocytes, i.e., individuals with low affinities or fitness values. In conventional neural network structures Artificial Immune System is used to minimize the final error but in this proposed method the error contributed by each neuron is calculated through back propagation from the Jacobian matrix. So, instead of having one fitness function for all the antibodies, we have as many number of fitness functions as number of neurons. In turn the cloning and hyper mutation of the antibodies, representing the weights and biases are dependent on the error contributed by the particular neuron.

A population of antibodies is initialized using binary strings each encoding weights and biases of the neural network. Affinity is calculated via fitness or objective values. Fitness is calculated as:

$$\text{fitness}^{j}(k) = 1/\left(1+e(t)^2\right) \qquad \ldots (1)$$

where e refers to one epoch error at time t. Error is calculated as:

$$e(t) = y(t) - d(t) \qquad \ldots (2)$$

Where $y(t)$ is the forecasted output and $d(t)$ is the desired output at time t. Each of the antibodies from the initial pool is copied into a number of clones to generate a temporary population of clones. Number of clones of each antibody is calculated as:

$$N * \left(fitness^{j}(k) \Big/ \sum_{i=1}^{N} fitness^{j}(i) \right) \quad \ldots (3)$$

where N is the number of antibodies of same type. $fitness(k)$ is the fitness function value of k^{th} antibody related to j^{th} neuron. This population of clones is made to undergo maturation process through hyper mutation mechanism. The hyper mutation is carried out via affinity based hyper mutation rate. Larger hyper mutation rate is set for lower affinity clones and vice versa. That is, the probability of hyper mutation of each clone is inversely to its affinity. In this case number of bits to be mutated is calculated as:

$$M * \left(\max\left(fitness^{j}\right) - fitness^{j}(k) \Big/ \max\left(fitness^{j}\right) - \min\left(fitness^{j}\right) \right) \ldots (4)$$

A new population of the same size as initial population of the antibodies is selected from the mutated clones and this completes the first iteration. In the next iteration, this fresh population is made to undergo cloning and hyper-mutation as discussed above and likewise.

The parameters of the system for next hour load forecasting are as given below:

MLP Neural Network Parameters

- No. of layers: 3 (Input layer, Hidden layer, Output layer)
- No of neurons in hidden layer: 4
- No of neurons in output layer: 1
- Activation function of hidden layer: logsig or tansig or tanh
- Activation function of output layer: Linear
- Training algorithm: Artificial Immune System
- No. of input variables: 9 (same day: hr-1, hr-2, hr-3; day-1: hr, hr-1, hr-2; day-7: hr, hr-1, hr-7)
- No of output variable: 1 (same day: hr)

- No of data sets in each generation: 21

Hybrid AIS Parameters

- Population size (N): 75
- Clonal population: $N * \left(fitness^{j}(k) \Big/ \sum_{i=1}^{N} fitness^{j}(i) \right)$
- Mutation probability: $M * \left(\max\left(fitness^{j}\right) - fitness^{j}(k) \Big/ \max\left(fitness^{j}\right) - \min\left(fitness^{j}\right) \right)$
- Number of generations: 20
- Antibody (M): 30 bit binary number
- Fitness function: 1 / (1+MSE2)
- No of cost function (s): 5, depending on the final output dependent error and error contributed by each hidden neuron. The error contribution of each hidden neuron is calculated by back propagating the final error.

Logsig, tansig, and tanh activation functions were taken for testing purpose. After much experimentation the MLP-NN and Hybrid-AIS parameters were selected as above. The MAPE obtained by using logsig, tansig and tanh activation functions were found to be 5.1523%, 4.9433% and 4.2036% respectively, as described in Table. 4.5. The corresponding results are shown in Fig. 4.7.(a),4.7.(c), and 4.7.(e). In Fig. 4.7.(b), 4.7.(d) and 4.7.(f) the graph shows the absolute mean square error for the given number of epochs. The forecasted load pattern is shown in dashed line and actual load pattern in solid line. Other activation functions did not significantly impact the load forecast accuracy.

The minimum Mean Average Percentage Error (MAPE) was found to be 4.2036 % by a network with tanh activation function (as shown in Fig. 4.7.(e)).

4.3.5 Simulation Results

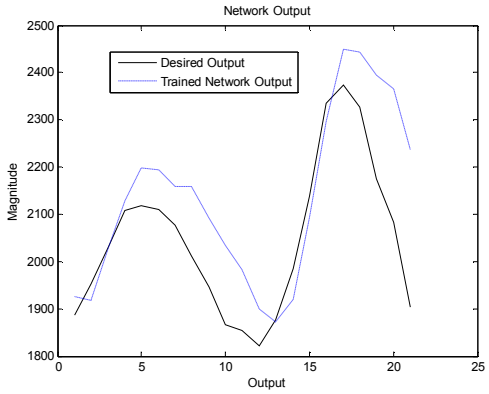


Fig. 4.7.(a) Logsig activation function

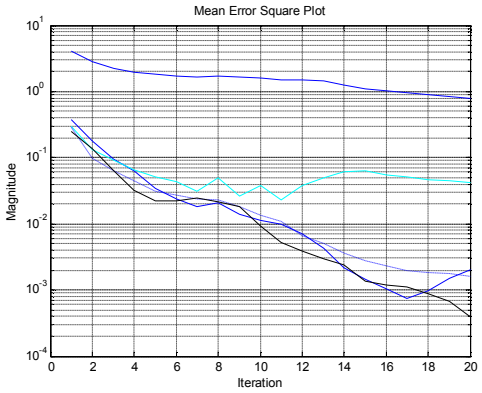


Fig. 4.7.(b) Logsig activation function

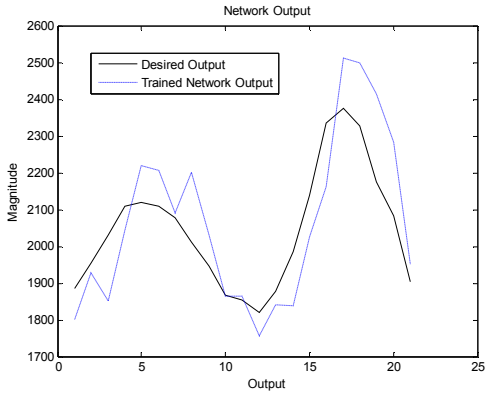


Fig. 4.7.(c) Tansig activation function

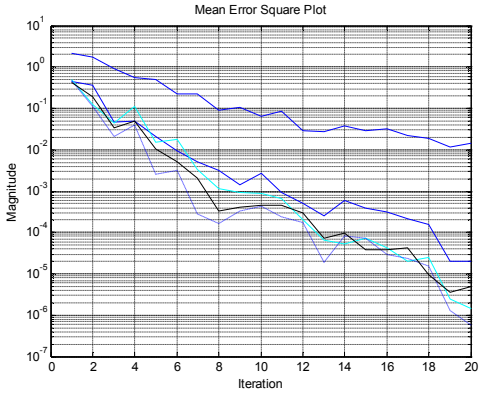


Fig. 4.7.(d) Tansig activation function

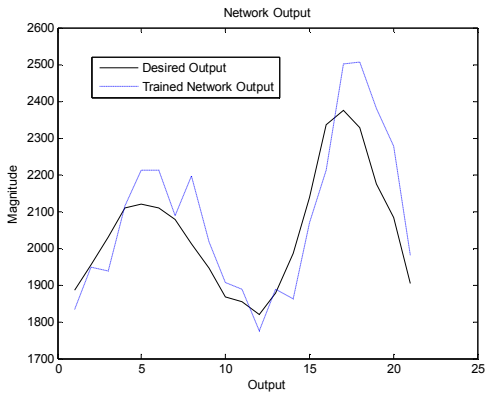


Fig. 4.7.(e) Tanh activation function

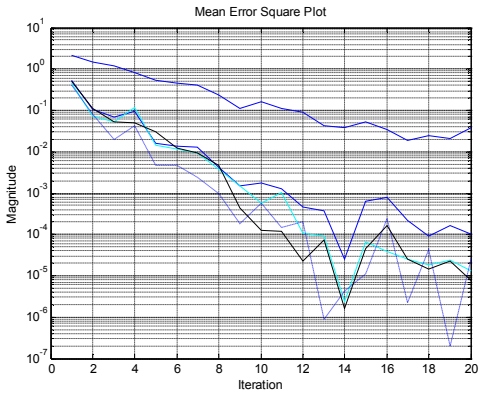


Fig. 4.7.(f) Tanh activation function

Fig. 4.7. Hour Ahead Prediction by MLPNN trained by AIS-BP Hybridized Algorithm with Different Activation Functions and Individual Fitness f^n [e = sum(error)]

tansig	logsig	tanh	
103.8101	108.199	88.2763	Total Abs. % Error
3	11	7	No. of Min. % Error
11	15	13	Negative % Error
6	7	8	Error less than 3%
4.9433	5.1523	4.2036	Avg. hourly % Error

Table 4.5. Performance Index Comparison for Hourly Load Forecasting by MLPNN-(BP –AIS Hybrid)

Chapter 5

CONCLUSION

5. Conclusion

5.1. Introduction

The main purpose of the thesis was to investigate application of various computational intelligence methods for short term load forecasting. In Chapter 3, STLF by three types of neural structures i.e. MLPNN, RNN and FLANN using back propagation training algorithm was discusses and their suitability for STLF was investigated. FLANN was modeled using Chebyshev, trigonometric, and algebraic expansion of input vector. RNN as proposed by Savaran [69] was modified to obtain better MAPE. Chapter 4 discusses about suitability of GA, PSO, and AIS as training algorithms for MLPNN and the corresponding suitability for STLF was investigated. GA and AIS were hybridized with back propagation to train the same neural structure and were found to be achieving better MAPE than utilizing only GA or PSO.

This chapter summarizes the work reported in this thesis, specifying the limitations of the study and provides some indications for future work. Following this introduction section 5.2 lists the achievements from the work. Section 5.3 provides the limitations and section 5.4 presents indications toward future work.

5.2 Achievement of the Thesis

From the simulation results as discussed in Table 5.1, it is understood that

1. Back Propagation hybridized GA and PSO trained MLPNN is capable of achieving better MAPE than utilizing only GA or PSO for training.

2. Modified RNN structure, as opposed to the RNN proposed by Savran, achieves the least MAPE, which in turn means the best of the results so far as all the models tested in this thesis is concerned.

From section 4.1, 4.2, and 4.3, it is understood that

3. MLPNN trained by GA or PSO required only 4 nos. hidden neurons, where as it required 17 hidden neurons in case of back propagation training. So, a leaner neural model trained by GA or PSO is capable of achieving by far the same accuracy as BP trained model.

4. GA and PSO trained networks require 150 iterations and 36 data sets, where as back propagation hybridized AIS require hardly 6 iterations and 21 data sets to converge to the same extent. This signifies its better convergence capability.

Learning Algorithms and Models		Hour Ahead MAPE in %	Hour Ahead MAPE Rank
MLP	3 Layer	3.1582	3
FLANN	Chebyshev	3.19	4
	Trigonometric	3.86	7
	Polynomial	3.1512	2
RNN	Modified Savaran Type	2.9633	1
MLP-GA	Single Activation Fn.	3.3909	6
	Individual Activation Fn.	3.1934	5
MLP-PSO		4.2118	9
MLP-AIS	Single Activation Fn.	5.2662	10
	Individual Activation Fn.	4.2036	8

Table 5.1. MAPE of Different Learning Algorithms and Models

5.3 Limitations of the Thesis

This thesis is prepared as an extraction of one year research work, which is part of Master of Technology curriculum.

1. Training and testing of all the models were conducted offline.
2. The hourly load data used was collected from one grid. The models developed herein need to be tested on data set from other grids, so that utility of these models can be verified for other load patterns. .
3. All the afore mentioned achievements of this thesis are concluded from simulations and experimentations. No mathematical justifications could be provided for the same.
4. Achievements of this thesis have to be verified in other fields of applications to make these models more robust and generalized.

5.4 Scope for Further Research

1. Lot many other feed-forward and feedback neural structures can be modeled for testing suitability of their use for STLF.

2. BFO and ACO can be used for training a MLPNN to find their capability for neural training.

3. Stability of the modified RNN has to be further investigated.

4. Hardware implementation of the implementations can be done to have a real time system.

DISSEMINATION

Following is the list of publications that has resulted from the work reported herein:-

National Conference Publications

1. "Short Term Load Forecasting using Neural Network trained with Genetic Algorithm", **Mishra, Sanjib**; Patra, Sarat Kumar; PCTCON 2008, Dindigul, Tamilnadu, India.

IEEE Indexed International Conference Publications

2. "Short Term Load Forecasting using Neural Network trained with Genetic Algorithm & Particle Swarm Optimization"
Mishra, Sanjib; Patra, Sarat Kumar; Emerging Trends in Engineering and Technology, 2008. ICETET '08, First International Conference on;16-18 July 2008 Page(s):606-611; ***Digital Object Identifier 10.1109/ICETET.2008.94*.**

3. "A Novel Approach to Design a Wireless Communication Based Railway Information System", Kumar, Vijay; Patra, Sarat Kumar; **Mishra, Sanjib**; TENCON 2008 - 2008, TENCON 2008. IEEE Region 10 Conference; 19-21 Nov. 2008 Page(s):1-4; ***Digital* Object Identifier 10.1109/TENCON.2008.4766653**.

4. "Short Term Load Forecasting using a novel Recurrent Neural Network"
Mishra, Sanjib; Patra, Sarat Kumar; TENCON 2008 - 2008, TENCON 2008. IEEE Region 10 Conference; 19-21 Nov. 2008 Page(s):1-6; ***Digital Object Identifier 10.1109/TENCON.2008.4766829***.

5. "Short Term Load Forecasting using a neural Network trained by a Hybrid Artificial Immune System"
Mishra, Sanjib; Patra, Sarat Kumar; ICIIS 2008, Third International Conference on Industrial and Information Systems, IEEE Region 10 Colloquium and Conference; 8-10 Dec. 2008 Page(s): 1-6; ***IEEE Catlog No.: CFP)0858A-CDR, ISBN: 978-1-4244-2806-9, Library of Congress: 2008906756.***

International Journal Publications

6. "Short Term Load Forecasting using a novel Recurrent Neural Network"
Mishra, Sanjib; Patra, Sarat Kumar; International Journal for Computational Intelligence, Theory & Practice, ***(In Press, Scheduled for Vol. 3, No 2, 2008).***

REFERENCES

[1] G. Gross, F. D. Galiana, 'Short-term load forecasting', Proceedings of the IEEE, 1987, 75(12), 1558 – 1571.

[2] A.D. Papalexopoulos, T.C. Hesterberg, 'A Regression Based Approach to Short Term Load Forecasting', IEEE Transactions on Power Systems, 1990, 5(1), 40 – 45.

[3] N. Amjady, 'Short-term hourly load forecasting using time-series modeling with peak load estimation capability', IEEE Transactions on Power Systems, 2001, 16(3), 498 – 505.

[4] W. Christianse, 'Short Term Load Forecasting Using General Exponential Smoothing', IEEE Transactions on PAS, 1971, 900 – 910.

[5] S.A. Villalba, C.A. Bel, 'Hybrid demand model for load estimation and short-term load forecasting in distribution electrical systems', IEEE Transactions on Power Delivery, 2000, 15(2), 764 – 769.

[6] J. Yang, H. Cheng, 'Application of SVM to power system short-term load forecast', Power System Automation Equipment China, 2004, 24(4), 30 – 32.

[7] Y. Li, T. Fang, E. Yu, 'Short-term electrical load forecasting using least squares support vector machines', International Conference on Power System Technology, 2002, 230 – 233.

[8] K.J. Hwan, G.W. Kim, 'A short-term load forecasting expert system', Proceedings of The Fifth Russian-Korean International Symposium on Science and Technology, 2001, 112 – 116.

[9] A.A. Desouky, M.M. Elkateb, 'Hybrid adaptive techniques for electric-load forecast using ANN and ARIMA', IEE Proceedings of Generation, Transmission and Distribution, 2000, 147(4), 213 - 217.

[10] K.H. Kim, H.A.Youn, Y.C. Kang, 'Short-term load forecasting for special days in anomalous load conditions using neural networks and fuzzy inference method', IEEE Transactions on Power Systems, 2000, 15(2), 559 – 565.

[11] R.F. Engle, C. Mustafa, J. Rice, 'Modeling peak electricity demand', Journal of Forecasting, 1992, 11, 241 – 251.

[12] O. Hyde, P.F. Hodnett, 'An Adaptable automated procedure for short-term electricity load forecasting', IEEE Transactions on Power Systems, 1997,12, 84 – 93.

[13] S. Ruzic, A. Vuckovic, N. Nikolic, 'Weather sensitive method for short-term load forecasting in electric power utility of Serbia', IEEE Transactions on Power Systems, 2003, 18, 1581 – 1586.

[14] T. Haida, S. Muto, 'Regression based peak load forecasting using a transformation technique', IEEE Transactions on Power Systems, 1994, 9, 1788 – 1794.

[15] W. Charytoniuk, M.S. Chen, P.Van Olinda, 'Nonparametric regression based short-term load forecasting', IEEE Transactions on Power Systems, 1998, 13, 725 – 730.

[16] J.Y. Fan, J.D. McDonald, 'A real-time implementation of short – term load forecasting for distribution power systems', IEEE Transactions on Power Systems, 1994, 9, 988 – 994.

[17] M.Y. Cho, J.C. Hwang, C.S. Chen, 'Customer short-term load forecasting by using ARIMA transfer function model', Proceedings of the International Conference on Energy Management and Power Delivery, EMPD, 1995, 1, 317 – 322.

[18] H.T.Yang, C.M. Huang, C.L. Huang, 'Identification of ARMAX model for short-term load forecasting, An evolutionary programming approach' IEEE Transactions on Power Systems, 1996, 11, 403 – 408.

[19] H.T. Yang, C.M. Huang, 'A new short-term load forecasting approach using self-organizing fuzzy ARMAX models', IEEE Transactions on Power Systems, 1998, 13, 217 – 225.

[20] M. Peng, N.F. Hubele, G.G. Karady, 'Advancement in the application of neural networks for short-term load forecasting', IEEE Transactions on Power Systems, 1992, 7, 250 – 257.

[21] A.G.Bakirtzis, et al., 'A neural network short-term load forecasting model for the Greek power system', IEEE Transactions on Power Systems, 1996, 11, 858 – 863.

[22] A.D. Papalexopoulos, S. Hao, T.M. Peng, 'An implementation of a neural network based load forecasting model for the EMS', IEEE Transactions on Power Systems, 1994, 9, 1956 – 1962.

[23] A. Khotanzad, et al., 'ANNSTLF - A neural-network-based electric load forecasting system', IEEE Transactions on Neural Networks, 8 (1997), 835 – 846.

[24] A. Khotanzad, R.A.Rohani, D. Maratukulam, 'ANNSTLF – Artificial neural network short-term load forecaster - Generation three', IEEE Transactions on Neural Networks, 1998, 13, 1413 – 1422.

[25] B.J. Chen, M.W. Chang, C.J. Lin, 'Load forecasting using support vector machines: A study on EUNITE competition 2001', http://www.csie.ntu.edu.tw/˜cjlin/libsvm. 2002.

[26] T.W.S. Chow, C.T. Leung, 'Nonlinear autoregressive integrated neural network model for short-term load forecasting', IEE Proceedings, Generation, Transmission and Distribution, 1996, 143, 500 – 506.

[27] S.E. Skarman, M. Georgiopoulous, 'Short-term electrical load forecasting using a fuzzy ARTMAP neural network', Proceedings of the SPIE, (1998), 181 - 191.

[28] Y. He, et al. 'Similar Day Selecting Based Neural Network Model and its Application in Short-Term Load Forecasting', Machine Learning and Cybernetics Proceedings of 2005 International Conference, 18 - 21 Aug. 2005, 4760 – 4763.

[29] K.L. Ho et al, 'Short-term load forecasting of Taiwan power system using a knowledge based expert system', IEEE Transactions on Power Systems, 1990, 5, 1214 – 1221.

[30] S. Rahman, O. Hazim, ‘Load forecasting for multiple sites: development of an expert system-based technique’, Electric Power Systems Research, 1996, 39, 161 – 169.

[31] S.J. Kiartzis, A. G. Bakirtzis, ‘A Fuzzy expert system for peak load forecasting. Application to the Greek power system’,10th Mediterranean Electrotechnical Conference, 2000, 3, 1097 – 1100.

[32] V. Miranda, C. Monteiro, ‘Fuzzy inference in spatial load forecasting’, Power Engineering Winter Meeting, IEEE, 2000, 2, 1063 – 1068.

[33] S.E. Skarman, M. Georgiopoulous, ‘Short-term electrical load forecasting using a fuzzy ARTMAP neural network’, Proceedings of the SPIE, 1998, 181 – 191.

[34] T. Hastie, R. Tibshirani, J. Friedman, ‘The elements of statistical learning: data mining, inference, and prediction’, Springer, New York, 2001.

[35] J. Han, M. Kamber, ‘Data Mining: Concepts and Techniques’, Morgan Kaufmann, San Francisco, California, 2001.

[36] H. Mori, and N. Kosemura, ‘Optimal regression tree based rule discovery for short-term load porecasting’, Proceedings of the IEEE Power Engineering Society Transmission and Distribution Conference, 2001, 1, 421 – 426.

[37] Y. Li, T. Fang, ‘Wavelet and support vector machines for short – term electrical load forecasting’, Proceedings of the International Conference on Wavelet Analysis and its Applications, 2003, 1, 399 - 404.

[38] Short term load forecasting using genetic algorithm and neural networks; Heng, E.T.H.; Srinivasan, D.; Liew, A.C.; Energy Management and Power Delivery, 1998. Proceedings of EMPD '98. 1998 International Conference on; Volume 2, 3-5 March 1998 Page(s):576 - 581 vol.2.

[39] Substation short term load forecasting using neural network with genetic algorithm; Worawit, T.; Wanchai, C.; TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering; Volume 3, 28-31 Oct. 2002 Page(s):1787 - 1790 vol.3.

[40] A time series approach to short term load forecasting through evolutionary programming structures; Chao-Ming Huang; Hong-Tzer Yang; Energy Management and Power Delivery, 1995. Proceedings of EMPD '95., 1995 International Conference on; Volume 2, 21-23 Nov. 1995 Page(s):583 - 588 vol.2.

[41] Short term load forecasting using genetically optimized neural network cascaded with a modified Kohonen clustering process Erkmen, I.; Ozdogan, A.; Intelligent Control, 1997. Proceedings of the 1997 IEEE International Symposium on; 16-18 July 1997 Page(s):107 – 112.

[42] BP-GA mixed algorithms for short-term load forecasting; YanXi Yang; Gang Zheng; Ding Liu; Info-tech and Info-net, 2001. Proceedings. ICII 2001 - Beijing. 2001 International Conferences on; Volume 4, 29 Oct.-1 Nov. 2001 Page(s):334 - 339 vol.4.

[43] Short-Term Load Forecasting Based on the Method of Genetic Programming; Limin Huo; Xinqiao Fan; Yunfang Xie; Jinliang Yin; Mechatronics and Automation, 2007. ICMA 2007. International Conference on ; 5-8 Aug. 2007 Page(s):839 – 843.

[44] Short Term Load Forecasting Using Particle Swarm Optimization Based ANN Approach; Azzam-ul-Asar; ul Hassnain, S.R.; Khan, A.; Neural Networks, 2007. IJCNN 2007. International Joint Conference on ; 12-17 Aug. 2007 Page(s):1476 – 1481.

[45] Short Term Load Forecasting Based on BP Neural Network Trained by PSO; Wei Sun; Ying Zou; Machine Learning and Cybernetics, 2007 International Conference on; Volume 5, 19-22 Aug. 2007 Page(s):2863 – 2868.

[46] Short-Term Load Forecasting Using Artificial Neural Network Based on Particle Swarm Optimization Algorithm; Bashir, Z.A.; El-Hawary, M.E.; Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference on; 22-26 April 2007 Page(s):272 – 275.

[47] Short-term load forecasting using artificial immune network; You Yong; Wang Sun'an; Sheng Wanxing; Power System Technology, 2002. Proceedings. PowerCon 2002. International Conference on; Volume 4, 13-17 Oct. 2002 Page(s):2322 - 2325 vol.4.

[48] Hybrid Neural Network Model for Short Term Load Forecasting; Yin, Chengqun; Kang, Lifeng; Sun, Wei; Third International Conference on Natural Computation, 2007.

[49] I. Erkmen, A. Oezdogan, 'Short Term Load Forecasting Using Genetically Optimized Neural Network Cascaded With a Modified Kohonen Clustering Process', Proceedings of the 12th IEEE International Symposium on Intelligent Control, 1997, 1, 107 – 112.

[50] D.P.Mandic, J.A.Chambers, Recurrent Neural networks for Prediction. New York: Jhon Wiley & Sons, 2001.

[51] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," Proc. Nat. Acad. Sci., vol. 79, pp. 2554–2558, 1982.

[52] J. L. Elman, "Finding structures in time," Cogn. Sci., vol. 14, pp. 179–211, 1990.

[53] M. I. Jordan, "Supervised learning and systems with excess degrees of freedom" COINS, Mass. Inst. Technol., Cambridge, MA, 1988, Tech. Rep. 88-27.

[54] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," Neural Comput., vol. 1, pp. 270–280, 1989.

[55] A. C. Tsoi and A. D. Back, "Locally recurrent globally feedforward networks: a critical reviewof architectures," IEEE Trans. Neural Netw., vol. 5, no. 2, pp. 229–239, Mar. 1994.

[56] S. A. Billings and C. F. Fung, "Recurrent radial basis function networks for adaptive noise cancellation," Neural Netw., vol. 8, no. 2, pp. 273–290, 1995.

[57] S. Santini, A. D. Bimbo, and R. Jain, "Block-structured recurrent neural networks," Neural Netw., vol. 8, no. 1, pp. 135–147, 1995.

[58] T. T. Lee and J. T. Teng, "The Chebyshev-polynomials-based unified model neural network for function approximation," IEEE Trans. Systems Man and Cybernetics, Part B, vol. 28, pp. 925-935, Dec. 1998.

[59] J. C. Patra and A. C. Kot, "Nonlinear dynamic system identification using Chebyshev functional link artificial neural networks" IEEE Tran. SMC, vol.32, pp. 505-51 1, Aug 2002.

[60] J. H. Holland, Adaptation in Natural and Artificial Systems, Ann Arbor, MI: Univ. Michigan Press, 1975.

[61] D. T. Pham and D. Karaboga, Intelligent Optimization Techniques, Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks. New York: Springer-Verlag, 2000.

[62] Z. Michalewicz, Genetic Algorithm+Data Structures=Evolution Programs, 2nd extended ed. New York: Springer-Verlag, 1994.

[63] H. Juidette and H. Youlal, "Fuzzy dynamic path planning using genetic algorithms, "Electron. Lett. vol. 36, no. 4, pp. 374-376, Feb. 2000.

[64] M. Setnes and H. Roubus, "GA-fuzzy modeling and classification: Complexity and performance, "IEEE Trans. Fuzzy Syst., vol. 8, pp. 509-522, Oct. 2000.

[65] X. Wang and M. Elbuluk, "Neural network control of induction machines using genetic algorithm training, "in Conf. Record 31st IAS Annual Meeting, vol. 3, 1996, pp. 1733-1740.

[66] L. Davis, Handbook of Genetic Algorithms. New York: Van Nostrand Reinhold, 1991.

[67] M. Srinivas and L. M. Pattanaik, "Genetic Algorithms: A survey, "IEEE Computer, vol. 27, pp. 17-27, June 1994.

[68] R. C. Eberhert and J. Kennedy, "A new optimizer using particle swarm theory", Proceeding of the Sixth International Symposium on Micro Machine and Human Science, page 39-43,, Nagoya, Japan, 1995, IEEE Service Center, Piscataway, Nj.

[69] A. Savran, Multifeeback-layer Neural Network, IEEE Trans. Neural Networks Vol. 18 No. 2, (2007) 373-384.

[70] J. H. Holland, Adaptation in Natural and Artificial Systems, Ann Arbor, MI: Univ. Michigan Press, 1975.

[71] D. T. Pham and D. Karaboga, Intelligent Optimization Techniques, Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks, New York: Springer-Verlag, 2000.

[72] L. Davis, "Handbook of Genetic Algorithms", New York: Van Nostrand Reinhold, 1991.

[73] R. C. Eberhert and J. Kennedy, "A new optimizer using particle swarm theory", Proceeding of the Sixth International Symposium on Micro Machine and Human Science, page 39-43,, Nagoya, Japan, 1995, IEEE Service Center, Piscataway, Nj.

[74] J.B. Park, K.S. Lee, J.R. Shin, K.Y. Lee, A particle swarm optimization for economic dispatch with non-smooth cost function, IEEE Trans. Power Syst. 20 (20) (2005) 34–42.

[75] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, second ed., IEEE Press, Piscataway, NJ, 2000.

[76] R.C. Eberhart, Y. Shi, Comparison between genetic algorithms and particle swarm optimization, in: Proc. IEEE Int. Conf. Evol. Comput., May, 1998, pp. 611–616.

[77] L. N. de Castro, F. J. Von Zuben, Artificial Immune Systems: Part II—A Survey of Applications. FEEC/Univ. Campinas, Campinas, Brazil.

[78] L.N. de Castro, F.J. Zuben, Learning and optimization using through the clonal selection principle, IEEE Trans. Power Syst. 6 (3) (2002) 239–251.

[79] A.S.Perelson and G. Weisbuch "Affinity maturation and learning in Immune Networks" in Molecular Evolution in rugged landscapes, pp. 189-205, ed. A Perelson and S. Kauffman, Addison Wesley, (1991).

[80] De Castro, L. N. & Von Zuben, F. J. (2000), "The Clonal Selection Algorithm with Engineering Applications", (full version, pre-print), In Proceedings of GECCO'00, Workshop on Artificial Immune Systems and Their Applications, pp. 36-37.